Laboratory for Parallel Programming

Bachelor's / Master's Thesis

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

Can LLMs Understand Program Logic?

Motivation

Large Language Models (LLMs) have emerged as candidate tools to tackle the automatic parallelization of source code as well as the generation of parallel code to solve a specific task at hand. While recent publications have shown promising results for this application of LLMs using various benchmark codes, the question whether they truly comprehend a programs logic or simply parrot what they have learned remains mostly uncovered. Although a potential parroting of learned examples could yield good results for small and simple problems, it is conceivable that it may fail if the complexity of the code or problem at hand increases due to inherent, characteristic problems of parallel programs such as data races. To obtain a better understanding of the capabilities of modern LLMs, it is crucial to answer the question mentioned above.

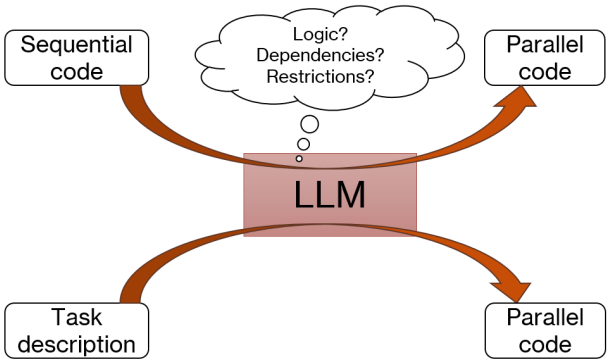

Illustration of exemplary code generation tasks to be examined.

Task

As a first step towards answering the general question, we will focus on the parallelization of sequential code. In this context, we will consider the knowledge of a programs control flow combined with the inherent data dependencies as a proxy for the programs logic. Your first task will be to conceive and develop approaches and benchmarks to assess the underlying understanding of program logic used by LLMs during the parallelization of sequential source code. One potential, exemplary approach could be to use chain-of-thought prompting to obtain insights into the program logic considered during the decision processes and compare these to the gathered information from an observed program execution via the DiscoPoP framework. Secondly, you will assess the boundaries of the LLMs capabilities and explore code features and characteristics that are beneficial/detrimental to the LLMs understanding of the program logic and, thus, their result quality in the parallelization use case.

Requirements

- Python, C/C++, (Clang/LLVM might be helpful, but not required)

Contact

Lukas Rothenberger <lukas.rothenberger@tu-darmstadt.de>

References

1. Daniel Nichols, Joshua H. Davis, Zhaojun Xie, Arjun Rajaram, and Abhinav Bhatele. *Can Large Language Models Write Parallel Code?* Association for Computing Machinery, New York, USA, 2024
2. DiscoPoP - Discovery of Potential Parallelism, *github.com/discopop-project/discopop*, Laboratory for Parallel Programming, TU Darmstadt

Issue date: **April 2025**