# Internship proposal

—

# Program Invariant Generation with a Large Language Model

**Keywords:** program verification; invariant generation; machine learning; large language model; automated deduction

## Description

Automated invariant generation is a key step towards the automatization of program verification. To verify that a program is correct (i.e., satisfies its specification), that program must be annotated with properties that remain true throughout its execution, the *invariants* [3]. This can be done manually, but it is a tedious and difficult task, it is thus necessary to automate it to open the way to better scaling and usability of verification techniques.

Large language models (LLM) have lately had a dramatic impact in the domain of language processing, with excellent results on varied tasks, including in the domain of programming languages [1]. The ability of LLM to produce statements from a given context seems well suited to producing program invariants [5], however LLM do not rely on language semantics, and it is therefore unlikely that they would produce correct invariants on their own. On the other hand, there exist tools for automated deduction (automated theorem provers, SMT solvers, etc.) that can be used to check whether a given formula is an invariant [6]. The aim of this internship is to combine the statistical approach of LLM and automated deduction techniques in a novel algorithm to analyze programs and generate their invariants. The LLM will generate candidates that will be checked by automated deduction. Correct invariants will be output, whereas others will lead to the production of a counter-example that will help the LLM improve its answer.

## Tasks

The intern will carry out the following tasks:

1. creating a set of training data consisting of annotated programs (with verification conditions and the invariants needed to verify them), e.g., based on programs used in automated verification competitions [4, 2];
2. choosing and training a LLM to generate invariants;
3. developing an algorithm that combines generation via LLM with automated deduction, in order to generate, verify and refine invariants;
4. evaluating the proposed algorithm.

## Skills

The candidate must be pursuing a master's degree in computer science or equivalent, and must have an interest in machine learning as well as in logic and program verification. Programming skills are necessary to implement the proposed solution. The working language can be either English or French.

## Practical Information

The internship will take place at LIRMM, the research department in computer science, robotics and microelectronics of the university of Montpellier, France. Its duration is five months. The internship is compensated according to the requirements of French legislation. The internship will be supervised by:


- Simon Robillard (research team MaREL) `simon.robillard@lirmm.fr`
- Maximos Skandalis (research team Texte) `maximos.skandalis@lirmm.fr`


## References


[1] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[2] Claire Dross, Carlo A Furia, Marieke Huisman, Rosemary Monahan, and Peter Müller. Verifythis 2019: a program verification competition. *International journal on software tools for technology transfer*, 23(6):883–893, 2021.

[3] Carlo A Furia, Bertrand Meyer, and Sergey Velder. Loop invariants: Analysis, classification, and examples. *ACM Computing Surveys (CSUR)*, 46(3):1–51, 2014.

[4] Vladimir Klebanov, Peter Müller, Natarajan Shankar, Gary T Leavens, Valentin Wüstholz, Eyad Alkassar, Rob Arthan, Derek Bronish, Rod Chapman, Ernie Cohen, et al. The 1st verified software competition: Experience report. In *International Symposium on Formal Methods*, pages 154–168. Springer, 2011.

[5] Kexin Pei, David Bieber, Kensen Shi, Charles Sutton, and Pengcheng Yin. Can large language models reason about program invariants? In *Proceedings of the 40th International Conference on Maching Learning*, 2023.

[6] Natarajan Shankar. Automated deduction for verification. *ACM Computing Surveys (CSUR)*, 41(4):1–56, 2009.