CoqPilot benchmarking framework

Gleb Solovev[1,3][0009-0004-1116-7743], Nikita Khramov[1,3][0009-0004-3968-4443], Andrei Kozyrev[1,3][0009-0004-3185-9368], and Anton Podkopaev[2,3][0000-0002-9448-6587]

first.last@jetbrains.com

[1] JetBrains Research, Germany
[2] JetBrains Research, Netherlands
[3] Constructor University Bremen, Germany

Abstract. We present CoqPilot benchmarking framework tailored for conducting the experiments involving LLMs and other techniques in Coq proof-generation domain. This framework allows for seamless setup of the experiment pipelines while ensuring the reliability and efficiency of their execution, making it suitable both for small preliminary studies and large-scale experiments. In addition to describing the benchmarking framework and its features, we present the experiment results obtained with it.

Keywords: LLM · Coq · code generation · benchmark

1 CoqPilot

CoqPilot [5] is a VS Code extension aiming to facilitate the process of writing Coq [3] proofs. It provides zero-setup experience and one-click generation, combining the capabilities of several tools. The main use case of CoqPilot is to complete the parts of the proofs marked with admit tactic. As the internals of CoqPilot are already described [5] and are not the focus of this work, we will not go further into the details of the plugin itself. Instead, we focus on CoqPilot benchmarking framework features and usage.

2 Benchmarking Framework

The main purpose of CoqPilot benchmarking framework is to create a convenient environment that allows researchers to experiment with different configurations, LLMs, and techniques for Coq proof generation.

We enable users to create flexible and reusable pipelines for the experiments. To introduce flexibility into the designing of the pipelines, we deliver domain-specific language (DSL) for defining experiment targets (e.g., theorems to prove) and models (configurable proof providers). The respective files are type-checked using Coq-LSP [1] to find target theorems and gather contextual information.

Combining all defined targets and models produces a matrix of *benchmarking tasks*. These tasks are executed asynchronously to optimize the performance.

One of the key features of the framework is its support for conducting both small experiments for preliminary studies and large-scale evaluations.

To ease the setup and execution of many small experiments during proof generation approaches testing, we introduce several helpful functionalities. First, the structured representations of type-checked files are cached to avoid the redundant type-checking. Also, we deliver a fail-fast strategy: in case of failure in one of the tasks, the whole pipeline terminates without waiting for other tasks to fail. Finally, the logging captures all relevant events that appeared during the execution of the pipeline in a real-time manner.

In the case of conducting large-scale experiments, reliability is crucial. Our framework ensures that no benchmarking results are lost or invalidated. Moreover, if the proof provider is temporally inaccessible (e.g., due to temporal tokens limits), the framework performs retries to acquire the proof eventually. It also includes a *resume-benchmarking* feature to continue the interrupted pipeline.

3 Experiments

We have conducted several experiments using `CoqPilot` benchmarking framework. We used the same data for the experiments as in [5]. We evaluated how many of the selected theorems could be proven using different LLMs [2] and methods [4], see Table 1.

Table 1. Benchmarking results

Reference proof length	$\leqslant 4$	$5-8$	$9-20$	Total
Group size	131	98	71	300
`firstorder auto with *`	11%	2%	1%	6%
OpenAI GPT-4o	50%	26%	15%	34%
OpenAI o1	66%	31%	8%	41%
Claude 3.5 Sonnet	73%	41%	27%	51%
LleMMa 7B	24%	11%	1%	15%
Tactician (`synth`)	45%	23%	10%	29%

Using the framework, we limited the number of *context theorems* (theorems with proofs sent to LLM as a few-shot prompt). The average number of context theorems without limitation is 42, and the ratio of proved theorems is 40%. We experimented with this parameter by using the OpenAI GPT-4o model on a dataset subset of 50 theorems. With only 7 context theorems, the ratio of proved theorems dropped to 28%. The framework also allows experimentation with different context theorem selection techniques. Using the technique measuring the similarity between target theorem and context theorem goals based on TF-IDF metric [6] instead of the default one based on the closeness to the target theorem, the ratio of proven theorems with only 7 context theorems raised to 38%.

References

1. Arias, E.J.G.: Visual studio code extension and language server protocol for coq (2022), https://github.com/ejgallego/coq-lsp
2. Azerbayev, Z., Schoelkopf, H., Paster, K., Santos, M.D., McAleer, S., Jiang, A.Q., Deng, J., Biderman, S., Welleck, S.: Llemma: An open language model for mathematics. arXiv preprint arXiv:2310.10631 (2023)
3. Bertot, Y., Castéran, P.: Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions. Springer Science & Business Media (2013)
4. Blaauwbroek, L., Urban, J., Geuvers, H.: The Tactician: A seamless, interactive tactic learner and prover for coq (2020). https://doi.org/10.1007/978-3-030-53518-6_17, http://dx.doi.org/10.1007/978-3-030-53518-6_17
5. Kozyrev, A., Solovev, G., Khramov, N., Podkopaev, A.: Coqpilot, a plugin for llm-based generation of proofs. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. pp. 2382–2385 (2024)
6. Sparck Jones, K.: A statistical interpretation of term specificity and its application in retrieval. Journal of documentation **28**(1), 11–21 (1972)