Short-Term Scientific Mission Grant
- APPLICATION FORM[1] -

Action number: CA20111

Applicant name: Maximilian Doré

Details of the STSM

Title: Conjecture and proof search in Agda with large language models

Start and end date: 28/01/2024 to 02/02/2024

Goals of the STSM

Purpose and summary of the STSM.

(max.200 word)

A dependently typed language such as Agda offers the possibility to write algorithms which are *correct by construction (c-by-c)* as the specification of a program can be encoded in its type, and the type checker then ensures that a program meets its specification. However, developing programs in such a way is a laborious endeavour as even for simple programs, the c-by-c version in Agda is very lengthy and intricate to encode.

The goal of this STSM is to leverage generative large language models (LLM) to help with developing c-by-c programs. We want to explore using LLMs in two ways: first, if one has an Agda program without expressive type, the LLM can be used to suggest a type that the program might satisfy. Second, after a more expressive type has been found, the program needs to be adapted to produce a proof that it satisfies the suggested type, for which we will again try to use an LLM.

These two working principles correspond to suggesting conjectures and finding proofs – two well-known problems in theorem proving – which will be studied in a novel framework: for a dependently typed programming language, and using an LLM.

Working Plan

Description of the work to be carried out by the applicant.

[1] This form is part of the application for a grant to visit a host organisation located in a different country than the country of affiliation. It is submitted to the COST Action MC via-e-COST. The Grant Awarding Coordinator coordinates the evaluation on behalf of the Action MC and informs the Grant Holder of the result of the evaluation for issuing the Grant Letter.

(max.500 word)

To kick off this project, we plan to work on the following points prior to and during the STSM:

- Curate a set of examples of Agda programs on which an LLM can be trained and tested.
- Try both proprietary LLMs and locally run LLMs based on the weights of a published foundational model. Proprietary LLMs will represent the state of the art but cannot easily be adapted for our purposes. Running our own LLM has the advantage that we can fine-tune the model and feed it training data specifically for our needs. In particular, we hope that Haskell programs provide good training data for LLMs as these are in effect less-annotated Agda programs.
- Integrate the LLM into Agda: the LLM should be callable through a tactic. The output of the LLM should be run through Agda's type-checker to catch ill-formed or ill-typed suggestions. In this case, the error messages of Agda should be fed back into the LLM, such that only sound expressions are given to the user.
- Develop a benchmark to gauge the effectiveness of the tool.

In order to curate a set of problems and gauge the effectiveness of our approach, we will compare our results with work that Moa Johansson has done on conjecture finding for Haskell programs, namely QuickSpec.

It will be particularly interesting to compare our results with work on using LLMs for the theorem prover Lean. Agda code is significantly more verbose than Lean's, as Lean heavily relies on tactics and thereby obfuscates crucial reasoning steps. We hypothesize that this verbosity makes it easier for LLMs to generate correct proofs in Agda than in Lean as the structure and workings of a proof are fully represented in the syntax.

Expected outputs and contribution to the Action MoU objectives and deliverables.

Main expected results and their contribution to the progress towards the Action objectives (either research coordination and/or capacity building objectives) and deliverables.

(max.500 words)

During the STSM, we hope to start working on the following deliverables:

- A test suite of simple Agda programs and their c-by-c elaborations.
- A large language model that is tailored to produce Agda code.
- An integration of a LLM as an Agda tactic.

We plan to further work out the deliverables over the rest of the year. We are planning to submit a publication describing our approach and results, for instance to the Conference on Certified Programs and Proofs 2024.

In terms of the research objectives of the EPN COST action, the project naturally contributes to point 6, developing the use of artificial intelligence and machine learning techniques on proofs. As our intended tool should be usable by non-experts in dependent type theory, we moreover hope to also contribute to point 3, making techniques for program verification more effective and more accessible. Large language models can also be characterised as search engines for unstructured data, if we manage to feed large proof libraries as training data into a LLM effectively we therefore also hope to contribute to point 5, providing tools for searching large libraries of formal proofs.

In terms of capacity building objectives of the COST action, we hope to increase the ease of access to formal verification techniques in education and other areas of science, namely in mathematics, which is point 4 of the COST objectives. Moreover, our project spans formal verification, formal methods and machine learning, thereby contributing to point 6, transferring knowledge in terms of expertise and scientific tools across the different disciplines.