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

Action number: CA20111

Applicant name: Matthew McIlree

Details of the STSM

Title: *Auditable Constraint Programming: Integrating Proof Logging with Fuzzing and Explanations*

Start and end date: *07/07/2024 to 20/07/2024*

Goals of the STSM

Purpose and summary of the STSM.

(max.200 word)

Constraint Programming (CP) solvers are powerful tools for automatically solving combinatorial problems. In many applications, it can be critically important to be able to trust that a solver returned the correct answer, and to detect and understand errors when they arise.

At the University of Glasgow, our research group led by Ciaran McCreesh has developed a prototype CP solver with support for proof logging [1]. This generates formal proofs of correctness alongside any results it obtains, which can be checked by an external program. We are aiming to foster further collaboration with a research group at KU Leuven, led by Tias Guns, whose work includes the development CPMpy: a solver-independent library for CP modelling in Python [2], and research into human-understandable step-wise explanations for CP models.

The aim of the proposed STSM is to leverage formal proof logs for better testing and auditing of solvers, ultimately leading to more trustworthy constraint programming. Specifically, our goal is to develop a CPMpy interface for the Glasgow Constraint solver and use this to investigate how proofs can be integrated into existing automated testing; using proofs for dynamic test generation ("fuzzing"); and understanding the relationship between formal proofs and stepwise explanations.

Working Plan

[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.

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

(max.500 word)

During my visit to KU Leuven, I would split my time between development of research software and discussions. The main topics to work on would be:

1. **Discussing and implementing a framework for handling solver-generated proofs within the CPMpy library**. We will have to consider how users can request and present proofs, and how reformulations of the user's specification might affect correctness. We will also consider how to interface with proof verification software to make the checking accessible.

2. **Developing an interface for the Glasgow Constraint Solver** [1] so that it can be included as a backend in CPMpy [2]. Substantial progress has already been made for this and so it should be possible to have version of this ready during the visit.

3**. Investigating how proof logs can be used as part of "fuzzing" for CP solver testing.** Fuzzing involves randomly generating inputs for a computer program (within certain parameters) and dynamically checking whether the program behaves as expected. We would look to improve existing fuzzing procedures with the assurances that proof logging provides for unsatisfiable problems and optimal objectives, as well as consider how to use the structure of proofs to generate better fuzz-test cases as part of a metamorphic testing procedure. It will also be interesting to consider how the certified result from a proof logging solver can be used to test or validate the results of other solvers.

4**. Discussing the relationship between formal proofs of correctness intended to be checked by software and step-wise explanations intended to be understood by humans.** This would be an initial discussion intended to foster further collaboration after the visit. We know already that formal proof logs, although explanatory in some sense, do not function as “good explanations” for the answer to the problem as understood in the context of explainable AI [3]. Likewise, step-wise explanation sequences are not intended to be machine checkable or provide a formal certificate of a solver’s result. Nevertheless, there are parallels to be drawn between the processes of generating proofs and generating explanations. Both require a faithful description of the problem being solved and identifying mathematical properties sufficient to justify derivations that eventually lead to the desired conclusion. Visualisation of solver-produced proofs would be one step towards understanding this relationship better, as would considering how we can translate between representations.

References:

[1] https://github.com/ciaranm/glasgow-constraint-solver

[2] https://cpmpy.readthedocs.io/en/latest/

[3] Bleukx, Ignace, et al. "Simplifying Step-Wise Explanation Sequences." *29th International Conference on Principles and Practice of Constraint Programming (CP 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

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)

By better integrating a proof generating constraint programming solver into an established modelling and testing library, within the accessible Python language, this STSM will contribute to the following MoU research co-ordination objectives of CA20111:

2. Promote the output of detailed, checkable proofs from automated theorem provers.

3. Make techniques for program verification more effective and more accessible to all stakeholders.

Given Python's strength in unifying powerful tools for different purposes including interfacing with machine learning and databases, it should also lay better foundations for future progress towards:

5. Provide tools for searching large libraries of formal proofs.

6. Develop the use of artificial intelligence and machine learning techniques on proofs.

Additionally by fostering further collaboration between Glasgow and Leuven and combining our scientific tools, it will clearly contribute to the capacity building objectives 1-8. Of particular emphasis would be:

4. Ease access to formal verification techniques in education and other areas of science.

since CPMpy is designed with ease of use and development of teaching material in mind, and so introducing formal proof logging into its ecosystem will enhance access to certification techniques for users outside of the formal verification community.

Our main expected results are a newly integrated solver backend for CPMpy with support for proof logging, a new set of accompanying proofs for test instances modelled or generated using Python, and a plan for future research directions in auditable constraint programming. We also expect that the visit will lay the groundwork for a tool-focussed publication that compares the efficacy of using proof-logging in conjunction with fuzzing for solver testing, relative to using either in isolation.

By developing an easier way to generate proofs and gaining a better understanding of how they can be useful, we promote the usability of verification methods and discover new challenges, tying into future deliverables of the action, particularly:

D11. Collection of verification challenges with summary of working recipes for verifying them.