Report on the outcomes of a Short-Term Scientific Mission[1]

Action number: CA20111

Grantee 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 15/07/2024

Description of the work carried out during the STSM

Description of the activities carried out during the STSM. Any deviations from the initial working plan shall also be described in this section.

NB: The STSM funded period was 07/07/2024 – 15/07/2024 but I was able to visit an extra five days with funding from another source.

The main work carried out during the STSM was completing the design and implementation of an interface between *CPMpy*, a system for modelling problems with constraints in Python developed at KU Leuven, and the *Glasgow Constraint Solver (GCS),* a constraint programming (CP) solver that certifies its results by producing machine-checkable proofs (proof logging). We created a simple functional API for using GCS as a backend solver for CPMpy, and for requesting, storing, and verifying the proofs it produces. It passes all unit tests and has been merged into the master branch of CPMpy's GitHub repository.

We also carried out a preliminary investigation into the integration of proofs with CPMpy's fuzz-testing framework. The current process involves selecting or generating a CP problem instance and verifying that certain properties (e.g. satisfiability) are maintained before and after certain mutations are randomly applied. We modified the procedure to check instead that solver still produces a valid proof, and noted that this allows for a much more diverse set of mutations to be applied, since regardless of the input problem, an incorrect GCS conclusion will always produce a failing proof. We were able to generate 44 failing proof instances with a prototype implementation, which indicate bugs in either the solving or proof-logging code for GCS. Of course, we also produced many more valid proofs, and these represent an additional benefit of the new fuzzer, since it gives us a way to quickly and easily generate a wide variety of correct proofs with different properties.

[1] This report is submitted by the grantee to the Action MC for approval and for claiming payment of the awarded grant. The Grant Awarding Coordinator coordinates the evaluation of this report on behalf of the Action MC and instructs the GH for payment of the Grant.

Funded by
the European Union

As another implementation task, we considered how to modify existing algorithms for finding “Minimal Unsatisfiable Subsets” of constraints (useful for explaining unsatisfiable CP problems) to instead find a minimal set of constraints for which the solver produces a failing proof. This would be useful for minimising bug instances as part of a delta-debugging framework that makes use of proofs.

The rest of the activities carried out during the STSM were mostly discussion-based. We considered how to fuzz-test the problem encoding required by GCS’s proof logging procedure, to increase trust that the proof matches the original problem that the user specified. We determined that it would be useful to have a mapping between proof variables and the original CP variables for this, opening up possibilities such analysing a failing proof to help extract a minimal bug instance, or analysing a valid proof to extract an explanation.

Finally, we created a plan for future research directions and collaboration. We committed to maintaining the new interface as part of CPMpy, and considered how we will use this to continue research into auditable constraint programming.

Description of the STSM main achievements and planned follow-up activities

Description and assessment of whether the STSM achieved its planned goals and expected outcomes, including specific contribution to Action objective and deliverables, or publications resulting from the STSM. Agreed plans for future follow-up collaborations shall also be described in this section

The STSM achieved its planned goals. The main achievements are:

1. The GCS-CPMpy interface, including the basic ideas for how proofs should be requested and verified from the python side and a plan for future project.
2. The proof-of-concept implementation for fuzz-testing with proofs.
3. Discussion of the relationship between problem reformulation and proof validity.
4. Discussion of the possibilities of fuzzing a proof-generating solver, and generation of both valid and valid proofs.
5. Discussion of the idea of fuzzing the encoding process of a proof logging procedure.
6. Discussion of the possibilities of using a proof-generating solver to help test other solvers, for example by treating it as a ground-truth to compare against, or verifying mutation properties themselves for standard fuzz-testing.
7. A short presentation given by me for the wider team working with Tias Guns at KU Leuven, on proof logging for CP and how the Glasgow Constraint Solver produces proofs.

These all contribute towards the objects of CAS20111, particularly

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.

We have agreed plans for follow-up collaboration, with an informal meeting at the 30th International Conference on Principles and Practice of Constraint Programming in September arranged in the first instance, as well as ongoing maintenance of the new interface via remote co-development. Going forward we hope to develop the fuzzing ideas into a tool-focussed publication bringing together proofs and automated test generation in the context of CP. This would contribute directly to the COST action D11.