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

Action number: CA20111

Applicant name: Jesper Amilon

Details of the STSM

Title: Automatic verification of floating-point programs using Constrained Horn Solvers.

Visitor: Jesper Amilon, PhD student at KTH Royal Institute of Technology, Sweden
Host: Prof. Philipp Rümmer, University of Regensburg, Germany

Start and end date: 11/09/2023 to 22/09/2023

Goals of the STSM

Purpose and summary of the STSM.

(max.200 word)

The overall goal of the visit is to discuss and plan for a framework for formal verification of floating-point programs, in the context of verification over CHCs (Constrained Horn Clauses).

Me (Jesper) and Prof. Rümmer have previously collaborated successfully, resulting in one published paper[1] and another one awaiting publication. Both works were centred around using the CHC-solver Eldarica[5], and its back-end SMT-solver Princess[6], for the purpose of automatically verifying and annotating C programs. Both tools are mainly developed at the University of Regensburg. With this visit, we intend to continue the collaboration by turning our focus to verification of floating-point programs. In particular, we plan to work on developing theories to support reasoning over real numbers in Horn Clause solvers. We shall consider such theories for program verification, but also for generating function contracts and loop invariants. Such annotations act as witnesses for program correctness and take the role of proofs.

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)

The plan for Jesper is to first learn, with help from the host, more about the implementation of Eldarica, in order to understand what theories and practicalities are needed to implement support for real arithmetic. This will also include learning about the underlying SMT-solver Princess used by Eldarica. Today, Eldarica has some, but limited, support for real numbers, due to a separate Bachelor thesis project, which we plan to build upon.

We shall first discuss how to finalise the current proof-of-concept implementation for floating point support in Eldarica; we then plan to go on to discuss how to use such an implementation for specification inference. Eldarica today has an existing framework for inference of ACSL[3] specifications, which we plan to extend to also cover floating-point programs. ACSL specifications can, e.g., be used for further verification of C programs in Frama-C[4].

An important part of the specification inference aspect is back translation of witnesses (proofs) of correctness over real numbers into contracts over floating-point programs. We will discuss how this back translation can be designed to encompass the differences between the domains of real numbers and floating-point numbers, such as the presence of round-off errors in the latter.

Another aspect we plan to consider in particular is a theory dedicated to handling transcendental functions, such as sine and cosine when verifying floating-point functions. Calls to such functions are ubiquitous in industrial embedded systems code (e.g., with calls to the `math.c` library for C programs). However, as pointed out, e.g., in [2], it is not well-understood how to handle such calls in program verification.

<u>Expected outputs and contribution to the Action MoU objectives and deliverables.</u>

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)

The expected output is a plan for how to handle industrial-scale programs containing floating-point programs in Horn-clause solvers, as well as to start an implementation that extends Eldarica for this purpose. We also plan to have an emerging approach for handling transcendental functions. With this work, we will mainly contribute to the working group 3 (Program Verification) of the COST action. Since the work touches upon SMT-solving, we may also contribute to working group 2 (Automated Theorem Provers).

First, the overall goal is the automated verification of floating-point programs, and thus we contribute to RCO3: *"Make techniques for program verification more effective and more accessible to all stakeholders."*

Within my (Jespers) research group, the main focus is on deductive verification of C programs, using high-level tools such as Frama-C. Philipp, on the other hand, has experience of developing techniques for, e.g., SMT-solving (Princess) and Horn-clause solving (Eldarica); thus, his knowledge will be a great resource for my work, since it can help for a better understanding of the back-end SMT-solvers for, e.g., Frama-C. This contributes to CBO1: "*Bring together members of the different communities working on proofs in Europe.*"

Also, my research project is in collaboration with the Swedish heavy-vehicle manufacturer Scania, and the idea for our work stems from the need of Scania to verify certain properties of floating-point programs of larger scale, related to the safety of the vehicles. Thus, our work also contributes to: CBO2: "*Act as a stakeholder platform in the field of formal proofs from its theoretical grounds to its industrial applications.*"

Lastly, the collaboration between my research group and Philipp has already seen several successes, and it is likely that it will continue so for the foreseeable future, contributing to CBO3: "*Create an excellent and inclusive network of researchers in Europe with lasting collaboration beyond the lifetime of the Action.*"

References

[1] Jesper Amilon et al. "Automatic Program Instrumentation for Automatic Verification (Extended Technical Report)". In: CoRR abs/2306.00004 (2023). arXiv: 2306.00004. url: https://doi.org/10.48550/arXiv.2306.00004.

[2] Roberto Bagnara et al. "A Practical Approach to Verification of Floating-Point C/C++ Programs with math.h/cmath Functions". In: ACM Trans. Softw. Eng. Methodol. 30.1 (2021), 9:1–9:53. url: https://doi.org/10.1145/3410875.

[3] P. Baudin et al. ACSL: ANSI/ISO C Specification Language. Url: http://frama-c.com/acsl.html. 2023.

[4] Pascal Cuoq et al. "Frama-C - A Software Analysis Perspective". In: Software Engineering and Formal Methods - 10th International Conference, SEFM 2012, Thessaloniki, Greece, October 1-5, 2012. Proceedings. Ed. by George Eleftherakis, Mike Hinchey, and Mike Holcombe. Vol. 7504. Lecture Notes in Computer Science. Springer, 2012, pp. 233–247. url: https://doi.org/10.1007/978-3-642-33826-7%5C_16.

[5] Hossein Hojjat and Philipp Rümmer. "The ELDARICA Horn Solver". In: 2018 Formal Methods in Computer Aided Design, FMCAD 2018, Austin, TX, USA, October 30 - November 2, 2018. Ed. by Nikolaj S. Bjørner and Arie Gurfinkel. IEEE, 2018, pp. 1–7. url: https://doi.org/10.23919/FMCAD.2018.8603013.

[6] Philipp Rümmer. "A Constraint Sequent Calculus for First-Order Logic with Linear Integer Arithmetic". In: Proceedings, 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning. Vol. 5330. LNCS. Springer, 2008, pp. 274–289. url: https://link.springer.com/chapter/10.1007/978-3-540-89439-1_20