

Interpolants from Clausal Proofs

Arie Gurfinkel¹ and Yakir Vizel²

¹ Carnegie Mellon Software Engineering Institute, Pittsburgh, USA
arie@cmu.edu

² Computer Science Department, The Technion, Haifa, Israel
yvizel@cs.technion.ac.il

Abstract

We present a method for interpolation based on DRUP proofs [1]. Interpolants are widely used in model checking, synthesis and other applications. Most interpolation algorithms rely on a resolution proof produced by a SAT-solver for unsatisfiable formulas. The proof is traversed and translated into an interpolant by replacing resolution steps with AND and OR gates. This process is efficient (once there is a proof) and generates interpolants that are linear in the size of the proof. As this method relies on the existence of a resolution proof, it depends on a SAT-solver that can log a resolution-proof.

In this work, we address three known weaknesses of this approach, due to its dependence on a resolution proof: (i) performance degradation experienced by the SAT-solver and the extra memory requirements needed when logging a resolution proof; (ii) the proof generated by the solver is not necessarily the "best" proof for interpolation, and (iii) combining proof logging with pre-processing is complicated. We show that these issues can be remedied by using DRUP proofs.

First, we show how to produce an interpolant from a DRUP proof, even when pre-processing is enabled. Then, we give a novel interpolation algorithm that produces interpolants partially in CNF. More precisely, our algorithm returns the CNF component of the interpolant. In case that the proof permits it, the interpolant is a pure CNF. Lastly, we show how a DRUP proof can be reconstructed on-the-fly to yield better interpolants. Our reconstruction aims at maximizing the CNF component of the interpolant forcing a specific type of resolutions.

We implemented our DRUP-based interpolation framework in MiniSAT, and evaluated its effect using AVY [2] - a SAT-based model checking algorithm that heavily relies on interpolation. We show the effect our method has on model checking and demonstrate the usefulness of our different heuristics.

References

- [1] Marijn Heule, Warren A. Hunt, Jr, and Nathan Wetzler. Trimming while checking clausal proofs. In *FMCAD*, pages 181–188, 2013.
- [2] Yakir Vizel and Arie Gurfinkel. Interpolating Property Directed Reachability. In *CAV*, 2014.