

Artifact for Interactive Bit Vector Reasoning using Verified Bitblasting

.1 Artifact check-list (meta-information)

A Introduction

This artifact contains the infrastructure and tooling necessary to display the performance of the verified bitblaster introduced by the paper for all the benchmarks presented.

- **Program:** The code repository for our framework along with the test suite. Note that this is already setup in the docker image.
- **Compilation:** The Lean4 toolchain, downloaded via `elan`. Note that this is already setup in the docker image.
- **Run-time environment:** Any operating system that supports Docker.
- **Hardware:** Any x86-64 machine.
- **Output:** Key theorems of the paper will be built and shown to have no unsound axioms.
- **How much disk space required (approximately)?:** 30GB
- **How much time is needed to prepare workflow (approximately)?:** 1hr
- **How much time is needed to complete experiments (approximately)?:** 5hr
- **Publicly available?:** Yes
- **Code licenses (if publicly available)?:** MIT
- **Archived (provide DOI)?:** 10.5281/zenodo.15755236

A.1 Performance

We test the performance of the verified bitblaster `bv_decide` against three benchmarks:

- `InstCombine` benchmark, extracted from LLVM's peephole verifier
- `HackersDelight` benchmark, containing bit-vector theorems extracted from the first two chapters of Hackers' Delight
- `SMT-LIB` benchmark, containing the problems from SMT-LIB's 2024 Competition

The artifact contains the benchmarks, the scripts to evaluate `bv_decide`'s performance, and the scripts to reproduce the plots in the paper.

B Hardware Dependencies

Podman or Docker are necessary to run our artifact (we tested it with the former). The container image has all dependencies needed to compile our framework with Lean4. **TODO: add info**

C Getting Started

Access the docker image from 10.5281/zenodo.15755236:

D Experiments Reproduction

Three main scripts are involved in the reproduction of our results and plots:

- `compare.py benchmark` runs our bitblaster as well as the solver we compare against for `benchmark`. The results obtained from this run are saved in `bv-evaluation/results`. Note that the number

Author's Contact Information:

of problems solved by `bv.decide` might slightly change depending on the performance of the machine in relation to the timeout set for the SAT/SMT solvers.

- `collect.py benchmark` collects and analyzes the results obtained for `benchmark` and stores everything in `bv-evaluation/raw-data`.
- `plot.py benchmark` plots the results obtained from `benchmark`'s run, including the plots presented in the paper.
- `collect-stats-bv-decide.py` collects all the statistics regarding our evaluation, and in particular the numbers we describe in the paper.

D.1 InstCombine - Reproduce Fig. 9

As an example, to reproduce Fig. 9 using 8 threads for the experimental run and 1 repetition, the sequence of commands to run are:

The expected plot is in `plots/bv.decide_stacked_perc_instCombine.pdf`