A Artifact Appendix

A.1 Artifact check-list (meta-information)

- **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.15754223

A.2 Description

A.2.1 Software dependencies. Docker is necessary to run our artifact. The Docker image has all dependencies needed to compile our framework with Lean4.

A.3 Experiment workflow

Access the docker image from 10.5281/zenodo.11519739.

```
$ docker load -i oopsla25-width-indep.tar
$ docker run -it siddudruid/oopsla25-width-indep
# | This clears the build cache,
# | fetches the maths library from the build cache,
# | and builds our framework.
$ cd /code/lean-mlir && lake clean && lake exe cache get && lake build
# | This allows to check that the key theorems of our framework are
# |guarded, and that they do not contain `sorry`s.
$ rg -g "SSA/Experimental/Bits/Fast/**/*.lean" "#guard_msgs in #print axioms" -C2 | grep "sorry"
$ rg -g "SSA/Experimental/Bits/Fast/**/*.lean" "#guard_msgs in #print axioms" -C2
# Run experiments, and check that the output is as expected.
$ /code/lean-mlir/run-experiments.sh
$ cat /code/lean-mlir/bv-evaluation/automata-automata-circuit-cactus-plot-data.tex
$ cat /code/lean-mlirSSA/Experimental/Bits/Fast/Dataset2/dataset2-cactus-plot-data.tex
```

A.4 Evaluation and expected results

On running lake build, the build succeeds with no errors. Next, run:

```
$ rg -g "SSA/Experimental/Bits/Fast/**/*.lean" "#guard_msgs in #print axioms" -C2 | grep "sorry"
```

Grep returns no matches, which checks that all guarded theorems do not use the sorry axiom. To manually inspect this, run:

```
$ rg -g "SSA/Experimental/Bits/Fast/**/*.lean" "#guard_msgs in #print axioms" -C2
```

In the output, observe the following lines, which tells us that, for example, the theorem denote_rewritePeepholeAt depends only on the axioms propext, Classical.choice, and Quot.sound. That is, this theorem *does not depend* on axioms such as sorryAx that compromise the correctness of Lean's verification. Now, since the Lean build was successful in the previous step, and since we use standard Lean axioms, we are justified in our claims that we have mechanized the key theorem statements from the paper.

Below, we list the key theorems that we claim to have mechanized, and their guarded verification below. Please inspect the output of ($ rg -g "**/*.lean" "#guard_msgs in #print axioms" -C2 | grep "sorry") and confirm that these guarded statements occur, with the guard containing only the axioms propext, Classical.choice, and Quot.sound.

A.4.1 Core Decision Procedure Theorems are Verified. The core correctness claim of the peephole rewriter is guarded below:

```
SSA/Core/Framework.lean
2422-
2423-/-- info: 'denote_rewritePeepholeAt' depends on axioms: [propext, Classical.choice, Quot.sound] -/
2424:#guard_msgs in #print axioms denote_rewritePeepholeAt
2425-
2426-/- repeatedly apply peephole on program. -/
--
```

```
2458-
2459-/-- info: 'denote_rewritePeephole' depends on axioms: [propext, Classical.choice, Quot.sound] -/
2460:#guard_msgs in #print axioms denote_rewritePeephole
2461-
2462-end SimpPeepholeApplier
```

A.4.2 Running The Solvers. Please run the folowing commands to run the solvers:

```
$ /code/lean-mlir/run-experiments.sh
```

*A.4.3 Verifying the Results for LLVM rewrites (*bv-evaluation*).* Please run:

```
$ cat /code/lean-mlir/bv-evaluation/automata-automata-circuit-cactus-plot-data.tex
```

This should produce the following output, where the timings might change depending on the machine that is run on. The absolute number of problems solved should be equal.

*A.4.4 Verifying the Results for the MBA dataset (*SSA/Experimental/Bits/Fast/Dataset2*).* Please run:

```
$ cat /code/lean-mlirSSA/Experimental/Bits/Fast/Dataset2/dataset2-cactus-plot-data.tex
```

This should produce the following output, where the timings might change depending on the machine that is run on. We expect the number of problems solved to be *at least* 1500 for KInduction, and all of 2500 for the other two solvers. The absolute number of problems solved can vary by 5-10 problems for KInduction, as the K-induction solver uses `cadical`, a SAT solver, which can have different timings and problems solved depending on the machine available.

A.5 Miscellanous Docker Usage

To copy files for inspection from the docker container into the host, keep the container running, and in another shell instance, use the `docker cp` command to copy files from within the container out to the host:[1]

```
$ docker container ls # find   ID
$ docker cp <CONTAINERID>:<PATH/INSIDE/CONTAINER> \
          <PATH/OUTSIDE/CONTAINER>
```

---

[1]For more about docker cp, please see: (https://docs.docker.com/engine/reference/commandline/cp/)