A Artifact Appendix

A.1 Abstract

We consider our artifact to be the *included version* of AWAMOCHE, as well as the tools and benchmarks we used in the paper. We stress that the results obtained for the same benchmarks by AWAMOCHE in the future might differ, as the tool will evolve.

The source code of AWAMOCHE is currently unstable. We will update its code and release a new version of the artifact along with the final version of our paper. We also plan to open-source AWAMOCHE as part of GENMC on Github.

A.2 Artifact check-list (meta-information)

- **Algorithm:** AWAMOCHE.
- **Program:** AWAMOCHE and C benchmarks.
- **Run-time environment:** Docker.
- **Output:** Console.
- **Experiments:** Scripts that fully reproduce the results are provided.
- **How much disk space required (approximately)?:** ~3 GB.
- **How much time is needed to prepare workflow (approximately)?:** Everything is already set up.
- **How much time is needed to complete experiments (approximately)?:** < 1h (see Section A.7).
- **Publicly available?:** Yes.
- **Code licenses (if publicly available)?:** See GENMC's webpage. For all code in the artifact not belonging to GENMC, GPLv2 applies.
- **Data licenses (if publicly available)?:** GPLv2.
- **Archived (provide DOI)?:** https://doi.org/10.5281/zenodo.7868371.

A.3 Description

A.3.1 How delivered.

The artifact (available on Zenodo) consists of a Docker image containing binaries for all the model checking tools used, along with all the benchmarks used in the submitted version of our paper, and AWAMOCHE's source code. These should suffice to validate the claims made in the paper.

A.3.2 Hardware dependencies.

None in particular; having at least 4GB RAM is recommended but not strictly required. Depending on your operating system, Docker might impose some extra hardware restrictions (see Docker's webpage).

A.3.3 Software dependencies.

An operating system on which Docker can be installed (see Docker's webpage) and Docker itself.

A.4 Installation

1. Download and install Docker, in case it is not already installed. On a Debian GNU/Linux distribution, Docker can be installed and started with the following commands:

   ```
   [sudo] apt install docker.io
   [sudo] systemctl start docker
   ```

   We have tested the artifact with Docker 20.10.23 under Debian GNU/Linux.
2. Next, import the Docker container containing AWAMOCHE:

   ```
   [sudo] docker import awamoche.docker awamoche
   ```

3. Finally, start up the image by issuing:

   ```
   [sudo] docker run -it awamoche bash
   ```

A.5 Experiment workflow

The results of the paper are reproduced using bash scripts that run benchmarks which validate our claims.

A.6 Paper claim-list

The most important claims made in the paper regarding the implementation of AWAMOCHE are summarized below:

1. **Section 6**: AWAMOCHE has exponentially better performance benefits than standard DPOR.
2. **Section 6:** AWAMOCHE applies not only to small synthetic benchmarks, but to realistic concurrent data structures as well.

Apart from these major claims above, minor claims regarding the tools' performance in particular benchmarks throughout Section 6. Since these claims easily follow from the tables/graphs reproduced as part of claims 1-2, we do not explicitly list them here.

A.7 Evaluation and expected result

In what follows, we assume that the working directory is `~/awamoche-benchmarks`. We provide a log file with the expected output for each script under `~/awamoche-benchmarks/logs`.

Columns starting with `E:` and `T:` show the number of executions explored and the time required by the corresponding tool, respectively. Executions are shown as the sum of complete and blocked executions.

As in the paper, we have set the timeout limit of 30m (denoted by *). This can be changed by appropriately setting L.41 in `get-table-data.sh`.

A.7.1 Reproducing Claims 1 and 2 (< 1h).

```
./get-table1.sh
```

This will run all tools on the benchmarks of Table 1. In contrast to the paper results, TruSt$_{\text{IPR}}$ performs slightly better than Trust$_{\text{STALE}}$ in `conf-loop`, `mpmc-queue` and `treiber-stack`. This is due to a minor configuration issue in the submitted version of the paper. The claim that TruSt$_{\text{IPR}}$ is inadequate to eliminate blocking (p.17) still holds. We will update the paper results.

A.8 Experiment customization

The artifact includes both AWAMOCHE and the various versions of TruSt (`trust`, `trust-stale`, and `trust-ipr`) used. AWAMOCHE is implemented on top of GENMC.

A.8.1 Running TruSt/AWAMOCHE.

A generic invocation of TruSt/AWAMOCHE looks like the following:

```
awamoche [OPTIONS] -- [CFLAGS] <file>
```

Where `CFLAGS` are options that will be passed directly to the C/C++ compiler, and `OPTIONS` include several options that can be passed to AWAMOCHE (`awamoche --help` prints a full list). `file` should be a C/C++ file that uses `pthreads` for concurrency.

More information regarding the usage of the tool, as well usage examples, can be found at GENMC's manual.

A.8.2 Available benchmarks.

The benchmarks we used for our paper are located under `~/awamoche-benchmarks/benchmarks`. Apart from the benchmarks located in the folder above, many more benchmarks can be found at GENMC's repository, an updated local copy of which (containing AWAMOCHE) can be found at `~/genmc-tool`.

In the above repository under `tests` (and the relevant sub-directories), there is a separate folder for each benchmark, that contains the "core" of the test case, as well as the expected results for the test case, some arguments necessary for the test case to run, etc. In order to actually run a test case, one can run the tool with one of the test case variants, which are located in a folder named `variants`, in turn located within the respective test case folder.

For example, to run a simple store buffering test with AWAMOCHE, please issue:

```
awamoche ~/genmc-tool/tests/correct/litmus/SB/variants/sb0.c
```

A.8.3 Code Layout.

AWAMOCHE's source code is located at `~/genmc-tool/src`. Important parts of the code pertinent to the explanations of Sections 3 and 4 of our paper can be found in the following files:

`GenMCDriver.{hpp,cpp}`: The main verification driver, incorporating AWAMOCHE (e.g., the function `revisitInPlace()` in-place-revisits a given read.)

`Interpreter.h, Execution.cpp`: A large part of the interpreter infrastructure.

`ExecutionGraph.{hpp,cpp}`: Default structure for an execution graph.

A more detailed explanation of GENMC's code layout can be found at this paper.