A Artifact Appendix

A.1 Abstract

This is the replication package for our paper "SPORE: Combining Symmetry and Partial Order Reduction". The artifact contains GenMC (which implements the TruSt algorithm) and Spore, as well as the tests used in the evaluation section of the paper.

Spore is publicly available on Github as part of GenMC. For any bugs, comments, or feedback regarding Spore, please send an email to `michalis@mpi-sws.org`.

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

- **Algorithm:** Spore.
- **Program:** Spore, GenMC and C benchmarks.
- **Run-time environment:** Docker.
- **Output:** Console.
- **Experiments:** Scripts that fully reproduce the paper's results are provided.
- **How much disk space required (approximately)?:** ~10 GB.
- **How much time is needed to prepare workflow (approximately)?:** Everything is already set up.
- **How much time is needed to complete experiments (approximately)?:** < 60h (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 or Spore, GPLv3 applies.
- **Data licenses (if publicly available)?:** GPLv3.
- **Archived (provide DOI)?:** 10.5281/zenodo.10798180.

A.3 Description

A.3.1 How delivered.

The artifact (available on Zenodo) consists of a Docker image containing Spore, GenMC, and the benchmarks used in the paper.

A.3.2 Hardware dependencies.

None in particular (having at least 8GB RAM is recommended but not strictly required; having multiple cores allows for benchmark parallelization). 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.

Note: Docker often performs poorly in Apple M1/M2 machines. Consider running Docker within a VM in such systems.

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 20.10.25 under Debian GNU/Linux.

2. Next, import the Docker container containing Kater:

```
[sudo] docker import spore.docker spore
```

3. Finally, start up the image by issuing:

```
[sudo] docker run -it spore 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 evaluation claims are summarized below:

1. **Section 4.1, Spore's scalability**: show that Spore scales well enough to verify useful implementations
2. **Section 4.1, external vs internal symmetries**: GenMC both types of symmetries are crucial in order to scale to more threads

Apart from these major claims above, minor claims regarding the tools' performance in particular benchmarks throughout the evaluation section. As these claims easily follow from the tables reproduced as part of claims 1-2, they are not explicitly listed here.

A.7 Evaluation and expected result

In what follows, we assume that the working directory is `~/spore-benchmarks`. Each subsection below first lists the command(s) that need to be run in order to validate the respective claim, and then provides some comments on the output.

The `execs`, `blocked`, and `time` columns show the number of full executions explored, the number of blocked executions explored, and the time required by the corresponding tool, respectively. A `nan` entry denotes a timeout. The timeout limit is adjusted as in the paper; it can be adjusted by setting the environment variable `TIMEOUT`.

A.7.1 Reproducing Claims 1-2 (< 60h).

The paper tables can be produced by running `get-tables.sh`. To parallelize benchmarking and speed up reproduction time, a numerical argument can be passed to the scripts. For example, to run all tables using 8 cores:

```
./get-tables.sh 8
```

The tables are printed in `stdout` and the `results` folder by default.

To produce the plots of the paper, you need to first run `./prepare-plots.sh`. You can then create the plots either by porting the resulting data along with `plot.tex` to your host machine (a copy of the data can also be found at `results/edited_data.txt`), or download LaTeX (including `texlive-pictures`), build `plot.tex` in the container, and then port/inspect `plot.pdf`.

One way to transfer files between the host machine the container is by using `-v`:

```
docker run -v /host/path/to/directory:/root/spore-benchmarks -it spore bash
```

This will map the host path `/host/path/to/directory` to the container's benchmarks folder.

A.8 Experiment customization

A.8.1 Running GenMC/Spore.

A generic invocation of GenMC/Spore looks like the following:

```
spore [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 GenMC (`genmc -help` prints a full list). `file` should be a C/C++ file that uses `pthreads` for concurrency.

More information regarding the usage of GenMC (as well usage examples) can be found at GenMC's manual.

The alias `trust` is provided to run a "pure" version of GenMC without Spore.

A.8.2 Available benchmarks.

The C benchmarks we used for our paper are located under `~/spore-benchmarks/benchmarks`. GenMC tests can be found at GenMC's repository, a modified local copy of which 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 GenMC, please issue:

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

A.8.3 Code Layout.

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

`GenMCDriver.{hpp,cpp}`: The main verification driver. Functions names ending in `SR` are related to SPORE.

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

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