Abstract

This artifact contains all the source codes and experimental data necessary to reproduce our evaluation in Section 5. In particular, the extended versions of the C11Tester [Luo and Demsky 2021] and GenMC [Kokologiannakis and Vafeiadis 2022] tools with our new algorithms, as well as all the benchmarks used in our evaluation are included. Python scripts that fully automate the process of reproducing our evaluation are also provided with the artifact.

Checklist

- **Algorithm:** Minimal coherence.
- **Program:** Extensions of GenMC and C11Tester with minimal coherence, and C/C++ benchmarks.
- **Run-time environment:** Docker.
- **Metrics:** Execution time, and number of executions explored.
- **Output:** CSV files.
- **Experiments:** Scripts that fully automate our workflow are provided.
- **How much disk space required (approximately)?:** ~7GB.
- **How much time is needed to complete experiments (approximately)?:** Reproducing all the results: ~60 hours (without parallelization). We also provide instructions for reducing the timeout durations and reproducing only a subset of our results.
- **Publicly available?:** Yes.
- **Code licenses (if publicly available)?:** GenMC is licensed under GPLv2 and C11Tester is licensed under MIT License. The remaining code in the artifact that is not part of these projects is licensed under MIT License.
- **Archived (provide DOI)?:** doi.org/10.5281/zenodo.7816526

Description

Hardware dependencies

Replicating the results of certain large benchmarks requires up to 32 GB RAM. Moreover, we remark that users may encounter problems in running the artifact on Apple M1 silicon due to certain incompatibility issues in running Docker on these machines. There are otherwise no special hardware requirements.

Software dependencies

The artifact requires a Docker installation.

Getting Started Guide

- Install Docker (https://www.docker.com).
- Obtain the artifact's docker image from doi.org/10.5281/zenodo.7816526.
- Import the image:

  ```
  docker import pldi23-179.docker pldi23-179
  ```

- Start a container:

  ```
  docker run -it pldi23-179 bash
  ```

- Compile the tools and benchmarks:

  ```
  sh /root/compile.sh
  ```

- Run basic tests:

  ```
  python3 /root/c11tester_evaluation/run.py -b sigma -i 1

  python3 /root/genmc_evaluation/run.py -b sigma -t 30s
  ```

- Compile the results:

  ```
  python3 /root/c11tester_evaluation/compile_results.py

  python3 /root/genmc_evaluation/compile_results.py
  ```

- If the installation was successful, the scripts will produce the following CSV files `/root/c11tester_evaluation/out.csv` and `/root/genmc_evaluation/out.csv` with relevant statistics.

  The outputs of these files can be displayed in the console as follows:

```
csvtool readable /root/c11tester_evaluation/out.csv

csvtool readable /root/genmc_evaluation/out.csv
```

Experiment workflow

Below we display the directory structure of the artifact. The two main directories are `c11tester_evaluation` and `genmc_evaluation`. Both of these directories are equipped with the scripts `run.py` and `compile_results.py` where the former is used to conduct the experiments and the latter is used to compile all the results into a CSV file.

```
├── root/
│   ├── c11tester_evaluation/
│   │   ├── benchmarks/
│   │   ├── c11tester/
│   │   ├── llvm/
│   │   ├── compile_results.py
│   │   ├── diff.txt
│   │   ├── run.py
│   ├── genmc_evaluation/
│   │   ├── genmc/
│   │   ├── compile_results.py
│   │   ├── diff.txt
│   │   ├── run.py
│   ├── compile.sh
│   ├── README.md
```

List of claims

The main evaluation claims of the paper are as follows.

- **Section 5.1, Stateless Model Checking Results (Table 1).** Replacing the consistency checking component of GenMC tool with minimal coherence speeds up the consistency checking procedure (measured wrt. execution time) and the overall model-checking task (measured wrt. number of executions explored).
- **Section 5.2, Online Testing Results (Table 2).** Replacing the consistency checking component of C11Tester tool with minimal coherence speeds up the consistency checking procedure (measured wrt. execution time).

Evaluation and expected result

Reproducing Table 1

The following command will run the original GenMC and minimal coherence extension of GenMC on all the benchmarks in Table 1 with 2 hours timeout per run.

```
python3 /root/genmc_evaluation/run.py
```

Note that the above procedure is expected to take ~50 hours on default settings. We refer the interested reader to the experiment customization section for instructions on reducing the timeout duration and/or performing the analysis on a select set of benchmarks. Once the experiments are finished, all the raw output logs will be located under the directory `/root/genmc_evaluation/outfiles`. Running the following script will process these outputs and produce a CSV file.

```
python3 /root/genmc_evaluation/compile_results.py
```

The default path of the generated CSV file is `/root/genmc_evaluation/out.csv`.

Reproducing Table 2

The following command will run the original C11Tester and minimal coherence extension of C11Tester on all the benchmarks in Table 2 with 10 iterations.

```
python3 /root/c11tester_evaluation/run.py
```

Note that the above procedure is expected to take ~5 hours on default settings. We refer the interested reader to the experiment customization section for instructions on reducing the number of iterations and/or performing the analysis on a select set of benchmarks. Once the experiments are finished, all the raw output logs will be located under the directory `/root/c11tester_evaluation/outfiles`. Running the following script will process these outputs and produce a CSV file.

```
python3 /root/c11tester_evaluation/compile_results.py
```

The default path of the generated CSV file is `/root/c11tester_evaluation/out.csv`.

Experiment customization

Running C11Tester on new benchmarks

Running C11Tester on a new C/C++ program requires instrumenting the program via C11Tester's LLVM pass. In the following, we provide instructions for achieving this and also demonstrate how the new program can be integrated into our experimental workflow. Running the below command will compile a program `<example_program>` with C11Tester's LLVM pass.

```
/root/c11tester_evaluation/benchmarks/clang++ <example_program> -o /root/c11tester_evaluation/benchmarks/example.out
```

After this step, the new program can be integrated into our experimental workflow as follows. The only required change is on the `get_benchmarks` function in the script `/root/c11tester_evaluation/run.py`. This function returns a list of tuples of shape `(benchmark name, path to binary)`. Users may extend this list with the new program (e.g., `(<example_program>, /root/c11tester_evaluation/benchmarks/example.out)`). After this step, the script `/root/c11tester_evaluation/run.py` can be utilized the same way as above.

Running GenMC on new benchmarks

Integrating a new program into our experimental workflow for GenMC can be achieved as follows. The only required change is on the `get_benchmarks` function in the script `/root/genmc_evaluation/run.py`. This function returns a list of tuples of shape `(benchmark name, path to source code)`. Users may extend this list with the new program (e.g., `(test, /root/genmc_evaluation/benchmarks/test.cpp)`). After this step, the script `/root/genmc_evaluation/run.py` can be utilized the same way as above.

Reducing timeout

In the experiments for reproducing Table 1, the timeout duration for each run can be altered by providing the argument `-t` to the script. For instance, the following sets the timeout to 10 minutes for each run:

```
python3 /root/genmc_evaluation/run.py -t 10m
```

Reducing iterations

In the experiments for reproducing Table 2, the number of iterations for each benchmark can be altered with by providing the argument `-i` to the script. For instance, the following sets the number of iterations to 3 for each benchmark:

```
python3 /root/c11tester_evaluation/run.py -i 3
```

Selecting benchmarks

The scripts `/root/genmc_evaluation/run.py` and `/root/c11tester_evaluation/run.py` both support running the corresponding analyses on a select set of benchmarks. This can be achieved by providing the argument `-b` and specifying the names of the focal benchmarks. For instance, the following runs the analysis on the benchmarks `control-flow`, `dekker`, and `sigma`.

```
python3 /root/c11tester_evaluation/run.py -b control-flow,dekker,sigma
```

Notes

We provide the diff files that demonstrate the modifications that we have performed on source codes of C11Tester and GenMC. These files can be found in `/root/c11tester_evaluation/diff.txt` and `/root/genmc_evaluation/diff.txt`, respectively. Interested readers may find further information on C11Tester and GenMC in their respective webpages:

- C11Tester (https://plrg.ics.uci.edu/c11tester/)
- GenMC (https://github.com/MPI-SWS/genmc)