HSCC 2022 Repeatability Evaluation instructions for "Multi-Requirement Testing using Focused Falsification"

1 Elements of the paper included in the REP

The REP contains MATLAB code to generate all the data in the paper, which is presented in five tables and one figure. An overview of the folder and file structure of the REP is given in the following list. Note that `main_REP.m` is the **only** file which needs to be edited and run to evaluate the REP; the rest of the files and folders are explained for reference only.

- `@MRF` contains a MATLAB class which is used to run Multi-Requirement Falsification (MRF) to generate data in the paper.
- `@MRFRunSummarizer` contains a MATLAB class which is used to take the data generated by the @MRF class and put the data into formatted Latex tables (.tex files).
- `ARCH_ATwSS` contains the benchmark model and specifications (from `https://github.com/decyphir/ARCH20_ATwSS`).
- `breach_modified` contains a modified version of the MATLAB toolbox Breach. For reference, Breach can be found at `https://github.com/decyphir/breach`. This modified Breach is provided to make sure that possible differences in the released version of Breach will not cause issues (this version is based on 1.9.0, but it is not identical).
- `FINAL_PAPER_TABLES` is a folder where the scripts will save final .tex files containing the tables that are in the paper.
- `results` is a folder where the scripts save data after each scenario is finished. The content of the files are explained by their naming – for example, the file `focused_snobfit_base_sim40_seed150.mat` was generated using the `focused_snobfit` solver (the main MRF algorithm) for the "base" parameter configuration, using 40 maximum simulations and the random seed 150.
- `scripts` contain different MATLAB scripts as well as some .mat-files with parameter settings to make the REP run smoothly.
- `specTransformer` contains the specification transformer that creates STL specifications from the Simulink specifications present in the `ARCH_ATwSS` folder. For reference, this is exactly the same code as provided at `https://github.com/JohanEddeland/specTransformer`.

An overview of the data generated by the REP is shown in Table 1.

2 System requirements for running the REP

2.1 Operating System

The software has been tested on Windows 10, Windows Server 2012, and Linux Ubuntu 20.04. It should also run fine on Mac OS with a working MATLAB installation and Intel processor.

Table 1: An overview of the data presented in the paper, and where it is generated in the REP.

Container	Description	Name of file in REP
Table 1	MRF performance metrics	`FINAL_PAPER_TABLES/headerTable.tex`
Table 2	MRF vs. Corners-Random summary	`FINAL_PAPER_TABLES/summarizerTable.tex`
Table 3	Results, non-artificial model, standard reqs	`FINAL_PAPER_TABLES/nonArtModel_standardSpecs.tex`
Table 4	Results, artificial model, standard reqs	`FINAL_PAPER_TABLES/artModel_standardSpecs.tex`
Table 5	Results, artificial model, artificial reqs	`FINAL_PAPER_TABLES/artModel_aSpecs.tex`
Figure 2	Sensitivity analysis data	`FINAL_PAPER_TABLES/sensitivityFigure.tex`
ALL	One document with all containers	`FINAL_PAPER_TABLES/fileWithAllTables.tex`

2.2 Software

MATLAB with Simulink is required. We have tested with MATLAB versions 2019b and 2020b. Any version after 2019b will probably work, but versions older than 2019b will not work since the Simulink models were saved using 2019b.

You may need to setup a compiler for MATLAB using the `mex -setup` command (in case recompilation of Breach binaries is necessary). We have used Microsoft Windows SDK 7 (`https://www.microsoft.com/en-us/download/details.aspx?id=8279`), and it works. **Note:** For Windows 10, you might have problems installing Windows SDK 7. For troubleshooting, go to the following link:
`https://se.mathworks.com/matlabcentral/answers/233850-how-can-i-install-sdk-7-1-on-windows-10`.

The results are generated as LaTeX tables (`.tex` files). To see them properly, you need to be able to generate PDF files from `.tex` files. If you do not have LaTeX installed, we recommend e.g. `https://www.overleaf.com/` as a quick way to get PDF files from the generated `.tex` files.

3 Instructions for installing and running the software

Note: To avoid any problems with conflicts in the MATLAB path (for example if one already has installed Breach och S-TaLiRo and they are on the MATLAB path), we recommend to temporarily clear the MATLAB path while running the REP (see item 2 in the list below).

The rest of the installation instructions are as follows:

1. Make sure MATLAB is installed.
2. To avoid MATLAB path conflicts, first save your current path using `savepath`, then restore the default MATLAB path using `restoredefaultpath`.

3a Run `main_REP.m` without changing any parameter values. **This is a minimal test run and should take at most 20 minutes to run.**

3b To run the full, real experiment with the data presented in the paper, change the parameters in `main_REP.m` to indicated values before running the file. **On a standard laptop, this will take more than a week to run.**.

4 To check the results from your run, compile the latex file `FINAL_PAPER_RESULTS/fileWithAllTables.tex` into a PDF. This file contains all tables and the figure as specified in Table 1.

Note that a run which is interrupted will still save intermediate data, so just running `main_REP.m` again will not rerun the scenarios that have already been generated (the generated data is stored in the `results` folder).

4 Details on the `main_REP.m` script

The original parameter values in `main_REP.m` only run a shorter version of the full experiment that have been run to get the data presented in the paper. To be specific:

1. The minimal run sets `totalMaxEval = 40`, while the experiments in the paper use `totalMaxEval = 3000`. This parameter sets the maximum number of simulations for each scenario.

2. The minimal run sets `allConfigurationFiles = {'all_base_v1.0.1.mat'}`, while the experiments in the paper use `allConfigurationFiles = {'all_base_artificial_v1.0.1.mat', 'all_hard_artificial_v1.0.1.mat', 'all_base_v1.0.1.mat', 'all_hard_v1.0.1.mat'}`. In other words, the minimal run only runs the scenario with base parameter values and non-artificial model, while the full experiments in the paper use all combinations of base/hard parameter values and artificial/non-artificial models.

3. The minimal run sets `allRandomSeeds = 150`, while the experiments in the paper use `allRandomSeeds = [150, 151, 152, 153, 154, 155, 156, 200, 201, 202]`. This is the list of seeds used for the experiments (minimal run uses one seed, paper uses 10 seeds).

4. Both the minimal run and the experiments in the paper set `allFalsificationModes = {'corners_pseudorandom', 'focused_snobfit'}`. One can technically only run one of these alternatives instead of both, but since having any sort of comparison requires both modes, we include both also in the minimal run.

4.1 Format of generated data

The data generated by the scripts are stored in `.mat` files in the `results` folder. Each file corresponds to a scenario and is named according to the convention `solver_difficulty_simX_seedY.mat`. Here, `solver` is `focused_snobfit` or `corners_pseudorandom`, `difficulty` is `base` or `hard`, and `X` and `Y` are the max number of simulations and random seed, respectively.

4.2 Generated files

`main_REP.m` generates several LaTeX files into the `FINAL_PAPER_TABLES` folder, as specified in Table 1. Note that the last entry of the table is a collection of all the tables and the figure in **one** LaTeX document. In other words. **in order to check the results from a run, one only needs to generate a PDF from the LaTeX document** `FINAL_PAPER_TABLES/fileWithAllTables.tex`.

5 Reusing Focused Falsification for other models and specifications

You can use Focused Falsification for other models and specifications if you wish. The following instructions are for the interested user only; they are not part of the repeatability process, instead they are put here to make it possible for users to use Focused Falsification in their own applications.

The main way of running Focused Falsification one time is seen in the file `runManyMRF.m`, on lines 47 - 61:

```
1  global BreachGlobOpt;
2  BreachGlobOpt.NormalizePredicates = 1;
3
4  mrfResults = MRF(modelToUse, totalMaxEval, B, R, ...
5                   currentReqs, thisFile, thisMode, thisSeed);
6
7  mrfResults.discreteValuedSignals = discreteValuedSignals;
8  mrfResults.parallelBatchSize = parallelBatchSize;
9  mrfResults.focusedReqSelectionMethod = 'lowestRobustness';
10 mrfResults.testStrategy = 'either';
11
12 mrfResults = mrfResults.run();
```

We now go through more details about each command in the code above, in order to facilitate use for other models than what is shown in the REP.

In line 1 and 2, a global variable is defined which enables predicate normalization (see paper for details). The global variable is used in the file `breach_modified/STL_Formula/private/generic_predicate.m`. In that file, there is a check of the global variable in lines 194-196, and if `NormalizePredicates` exists as a field in `BreachGlobOpt`, the function `normalize_rob` on lines 202 - 212 is called to normalize the predicate robustness values.

To run the `MRF` class instantiation on lines 4 - 5, one needs the following:

- `modelToUse`: This is the name of a Simulink model. For example, if you have a model called `myModel.slx`, you would define `modelToUse = 'myModel'`.
- `totalMaxEval` is the maximum number of simulations to use (total simulation budget).
- `B` is a BreachSimulinkSystem. For details on how to create and modify a BreachSimulinkSystem object, see the README in `https://github.com/decyphir/breach`.
- `R` is a BreachRequirement object with all the requirements one wants to test. For details on how to create and modify a BreachRequirement object, see the README in `https://github.com/decyphir/breach`.
- `currentReqs` is a cell array containing `STL_Formula` objects. `STL_Formula` objects are created either directly through the constructor, or by reading `.stl` files with the `STL_ReadFile` function. Note that `R` and `currentReqs` are usually related as simply as `R = BreachRequirement(currentReqs)`.
- `configurationFile` is just the name of the file used for the REP, and this name is only used to name results variables accordingly. The value does not matter for running the Focused Falsification algorithm, i.e., one could set `configurationFile = 'dontCare'`.
- `thisMode` should be either `'corners_pseudorandom'` for running the baseline comparison algorithm, or `'focused_snobfit'` for running the Focused Falsification algorithm (using the SNOBFIT solver for the focused phase as explained in the paper). `thisSeed` is the seed to use (an integer). This is used to guarantee reproducibility of results.

In line 7, one can define the optional cell array of discrete-valued signals in the model (for example `discreteValuedSignals = {'signal1', 'signal2', 'myOtherSignal'}`). **Note that this is not important to use if you use the standard setting of** `mrfResults.focusedReqSelectionMethod = 'lowestRobustness'`.

In line 8, the parallel batch size is used (this is used by the inner `FalsificationProblem` object). Since we do not recommend using parallel computations right now, this argument does not matter (set it to e.g. 50).

We recommend setting lines 9 and 10 according to what we show here, but you should be aware that there are other possibilities to use if you want to experiment.

Line 12 runs the actual Focused Falsification. Afterward, you can access different types of interior results in the `mrfResults` object, and the results are also saved to a file (similar to what is shown earlier in the REP).