ExLi Appendix

Automatically extracting inline tests from unit tests.

Introduction

This repo contains the code and data for producing the experiments in ExLi paper.

Inline test format

1. "Declare" part `itest`

In our experiments, we use `itest(test_source, target_stmt_line_number)`, to represent the test source and the line number of the target statement in original file (note that the original file is different from the Java file with inline tests). For example, `itest("Randoop", 57)` means that the test source is Randoop generated tests, and the target statement starts at line 57 in the original file.

2. "Assign part" part `given(var, value)`
3. "Assert" part `assert(var, value)`

How to use ExLi

Install

Build a docker image

`docker build -t exli .`

`docker run -it exli /bin/bash`

In the docker, create a Python environment named `exli`

`cd exli/python && bash prepare-conda-env.sh`

`conda activate exli`

Usage

Generate unit tests and inline tests

In `exli/python` directory

(Optional) Find the target statements. It will help EvoSuite reduce the search scope. Otherwise, EvoSuite will generate tests on the whole project. The generated target statements are in `results/target-stmt/Bernardo-MG_velocity-config-tool-26226f5.txt`

`python -m exli.main find_target_stmts --project_name=Bernardo-MG_velocity-config-tool --sha=26226f5 --target_stmts_path=${HOME}/exli/results/target-stmt/Bernardo-MG_velocity-config-tool-26226f5.txt`

Alternatively, to use the default setting for output file

`python -m exli.main batch_find_target_stmts --test_project_name=Bernardo-MG_velocity-config-tool`

`python -m exli.main run --project_name=Bernardo-MG_velocity-config-tool --sha=26226f5 --randoop=True --randoop_tl=100 --unit=True --evosuite=True --evosuite_tl=120 --seed=42 --log_path=${HOME}/exli/log/raninline.log`

Alternatively, to use the default setting for test generation and output dirs

`python -m exli.main batch_run --test_project_name=Bernardo-MG_velocity-config-tool`

The generated inline tests are in

`all-tests/Bernardo-MG_velocity-config-tool`

Execute the generated inline tests

In `exli/python` directory

`python -m exli.main run_inline_tests --project_name=Bernardo-MG_velocity-config-tool --sha=26226f5 --generated_tests_dir=${HOME}/exli/reduced-tests/Bernardo-MG_velocity-config-tool-26226f5 --inline_tests_dir=${HOME}/exli/reduced-its/Bernardo-MG_velocity-config-tool-26226f5 --inlinetest_report_path=${HOME}/exli/results/reduced-its-report/Bernardo-MG_velocity-config-tool-26226f5.json --cached_objects_dir=${HOME}/exli/all-tests/Bernardo-MG_velocity-config-tool-26226f5/.inlinegen --deps_file=${HOME}/exli/generated-tests/Bernardo-MG_velocity-config-tool-26226f5/deps.txt --parse_inline_tests=True --`

`log_path=${HOME}/exli/log/run-its.log`

Alternatively, to use the default setting for output dirs

`python -m exli.main batch_run_inline_tests --test_project_name=Bernardo-MG_velocity-config-tool`

If there are failed inline tests, run the following command to remove the failed inline tests

```
python -m exli.main analyze_inline_tests_reports --inline_test_type=reduced

python -m exli.main analyze_inline_tests_reports --inline_test_type=all

python -m exli.main remove_failed_tests --inline_test_type reduced

python -m exli.main remove_failed_tests --inline_test_type all

python -m exli.main batch_run_inline_tests --test_project_name=Bernardo-MG_velocity-config-tool # re-generate test report
```

The generated execution result can be found at

`results/all-its-report/Bernardo-MG_velocity-config-tool-26226f5.json`

Generate mutants and run mutation analysis

In `exli/python` directory

`python -m exli.eval batch_run_generate_mutants --test_project_name Bernardo-MG_velocity-config-tool`

`python -m exli.eval batch_run_tests_with_mutants --test_project_name Bernardo-MG_velocity-config-tool`

Repo structure

raninline: This directory constains the source code of TargetStmtFinder, VariablesFinder, Instrumenter, Collector, Round1Reducer, and InlineTestConstructor.

Generated tests

R0-tests, R1-tests, R2-tests directories contain the inline tests that are integrated with source code.

R0-tests: It contains the R0 tests.

R1-tests: It contains the R1 tests.

R2-tests: It contains the R2 tests.