How to set-up the experiment ---- details

# There are two main steps to do the experiment: prepare the subject, conduct the experiment.

Prepare the subject.

1. Exhaustively run the test sets

For each possible configuration, run the test case under the configuration and collect the outcomes. Then , obtain the mapping collection: test configuration 🡪 outcome.

For example, for the HSQLDB 2.rc8, the result should be collected as follows (Note that we test 8 options on this version, the test model is 3×2×2×2×2×2×2×2×4×3×2×2):

0 0 0 0 0 0 0 0 0 0 0 0 🡪 pass …...

1 0 0 0 1 0 1 0 0 0 0 1 🡪 exception1

……

1 0 1 0 1 0 1 0 2 0 0 1 🡪 exception2

…...

2 1 1 1 1 1 1 1 3 2 1 1 🡪 pass

Specifically, for different subject, this preparation is as follows:

* HSQLDB: Import the test 2rc8, test2.25, and test 2.29, into the eclipse as a project. Each subject has one test case which are, *test2rc8 – scr/org.hsqldb.test/ScrollAndLongString.java, test2.25 -- scr/org.hsqldb.test/IncompatibleDataAndUpdateRow.java, and test2.29 -- scr/org.hsqldb.test/TestJoinAndDelete.java*, respectively. Run the main testing function (*test2rc8 – scr/org.hsqldb.test/TestScrollAndLong.java, test2.25 -- scr/org.hsqldb.test/TestIncompatibleDataAndUpdateRow.java, and test2.29 -- scr/org.hsqldb.test/JoinAndDeleteRows.java*), which test the test case under each possible configuration of that subject. The results will show into file “resultNew.txt”, and note that we use different number to represent different fault a configuration triggers, the mapping of number to specific exception information is listed in file “bugInfoNew.txt”.
* JFlex: Import the Jflex1.4.1, Jflex1.4.2 into the eclipse as a project. The test case of each subject is based on the text files under each subject (For Jflex1.4.1, they are State.jflex, State\_options.jflex, State\_normal.jflex, State\_normal\_options.jflex, Bug.jflex, Bug\_options.jflex, Bug\_add.jflex, and Bug\_add\_options.jflex , while for Jflex 1.4.2, they are testF.jflex, testF\_options.jflex,remove\_type.jflex,remove\_type\_options.jflex, Bug2.jflex, Bug2\_remove.jflex, Bug2\_remove\_options.jflex, and Bug2\_options.jflex ). Run the main testing function (*Jflex1.4.1– scr/org.jflex.test/TestJFlex.java, Jflex1.4.2-- scr/org.jflex.test/TestJFlex.java*). The results will show into file "result\_of\_testCase.txt", and the mapping of number to specific exception information is listed in file “bugInfo.txt”.
* Grep: Import testGrep into the eclipse as a project. The test case of each version is listed in src/runGrep/GrepTestCase.java (function testBoth33080\_28588(int[] test) for version 2.22, and testBoth7600\_29537(int[] test) for 2.6.3 ). Run the main fuction (src/runGrep/BenchTestGrep2\_22.java for version 2.22 and src/runGrep/BenchTestGrep2\_6\_3.java for 2.6.3), and it will execute all the test cases for corresponding subjects. Note that, the corresponding Grep version should be installed first to run these functions.
* Synthetic: Import testSyn into the eclipse as a project. Run the main function (src/syn/ExpriSetUp), which will run each test cases of the five synthetic subjects and return the results (will be listed in result.txt).

1. Get the real MFS of each subject.

Through inspecting the source code and the bug tracker information (which is given in the readme.txt), we get the MFS of each subject. Specifically, they are as following:

• HSQLDB: 2rc8 --- MFS of fault 1: (- - - - - 1 0 0 - - - -), MFS of fault 2: (- - - - - 1 - - 2 2 - -)(- - - - - 1 -

- 2 1 - -) , MFS of fault 3: ( - - - - - 1 - - 3 2 - -) (- - - - - 1 - - 3 1 - -)

2.25 --- MFS of fault 1: (- - - - - - 1 0 - - -), MFS of fault 2: ( - - - - - 2 - - - - -).

2.29 --- MFS of fault 1: (- - - - - - 1 - - - -), MFS of fault 2: 1 - - - - 1 - 0 - - -)(0 - - - - 1 - 0 - - -) , MFS of fault 3: ( - - - - - 1 - 0 - - -)

• JFlex: 1.4.1 --- MFS of fault 1: (0 - - - - - - - - - - - -) , MFS of fault 2: (- 0 - - - - - - - - - - -).

1.4.2 --- MFS of fault 1: (- 0 1 - - - - - - - - - - -), MFS of fault 2: (1 - - - - - - - - - - - - -).

• Grep: 2.6.3 --- MFS of fault 1 (0 - - - - - -), MFS of fault 2 (- 1 1 - - - -), MFS of fault 3 (- - - 0 1 -

-), MFS of fault 3 (- - - 1 1 - -), MFS of fault 3 (- - - 2 1 - -),

2.22 --- MFS of fault 1 (0 - - - - - -), MFS of fault 2 (- 0 3 - - - -), MFS of fault 2 (- 1 3 - - - -),

MFS of fault 3 (- - - 0 0 - -).

• Synthetic:

1 --- MFS of fault 1: (-, -, 0, -, -, -, -, 0, -, -) MFS of fault 2: (-, -, -, 1, -, 1, -, -, -, -) MFS of fault 2: (-, -, -, -, 0, -, -, -, -, -), fault 3: (-, -, -, -, -, -, 0, 2, -, -) fault 4: (-, -, -, -, -, -, 1, -, 2, -)

2 --- MFS of fault 1: (-, -, 0, 0, -, -, -, -, -), MFS of fault 1: (-, -, 0, -, -, 1, -, -, -), MFS of fault

2: (-, -, -, -, 0, -, 1, -, -) MFS of fault 2: (-, -, -, 1, -, -, 0, -, -) MFS of fault 3: (-, -, 2, -, 3, -, -, -, -)

3 --- MFS of fault 1: (0, 0, -, -, -, -, -, -), MFS of fault 2: (-, 1, 1, -, -, -, -, -), MFS of fault 2: (-, -, 1, 1, -, -, -, -), MFS of fault 3: (-, -, -, -, 1, -, -, 1), MFS of fault 4: (-, -, -, -, -, 2, 2, -).

4 --- MFS of fault 1: (0, -, -, -, -, -, -), MFS of fault 2: (-, 1, -, 0, -, -, -), MFS of fault 3: (-, -, 1, -, -, -, -), MFS of fault 4: (-, -, -, -, 0, 0, -), MFS of fault 2: (-, -, -, -, -, -, 0)

5--- MFS of fault 1: (0, -, -, -, -, -, -), MFS of fault 2: (-, -, 0, 0, -, -, -), MFS of fault 2: (-, -, 1, -, 1, -, -), MFS of fault 3: (-, 2, 2, -, -, -, -), MFS of fault 4: (2, -, -, -, -, -, 0),

Conduct the experiment

For each failing test case obtained by the previous step, we run the following approaches to identify the MFS: FIC\_BS with two strategies (Regarded as the same failure, distinguishing failures) and Replacement strategy with two techniques (ILP and random).

Approach FDA-CIT does not work on single failing test case, instead, it works on covering arrays in an iterative way. Hence, a covering array generator is needed to generate the covering array. Then, for the first covering array it generated, use ILP to identify the MFS in these failing test cases in it (ignore the ones that contain the identified MFS). As for FDA-CIT, let the first covering array as a seed, and generate a t+1-way covering array. Then follow the FDA-CIT process, i.e, identify the MFS, check whether tested t-way coverage is satisfied, and go on generating test cases.

After those steps, the experiment can be easily conducted, make these approaches identify the MFS in the subjects, and compare them with real MFS we obtained previously.