1 Bug Localisation Framework: Options Synopsis

Usually the `--prepare` and `--mine` and `--class-sampler` (for mining on class level), and maybe `--DOTwriter`, is all that is needed. They serve as a convenient interface to the framework, as they execute several tests at once and use the following environment variables to generated parameters for the tools:

- `MINING_OUT` where all the data will be stored
- `IBUGS_DIR` iBUGS directory
- `JAVA_1.4` JRE v1.4 compiler and VM

Not each possible combination of options is checked for validity. The general form for executing `tools.jar` is: `<options> [data ]*`

1.1 `--prepare`

Prepares everything for further steps. First the tests associated with the given big ID are copied from the post-fix version to pre-fix version. Then the `.js` files within `<iBUGS dir>/instrumentation/lib/js_variations` are used to generate variations of those tests (usually just a single one). After that we generate a list file containing tests with a likelihood of `-l` percent. All tests supposed to fail will be included. Then the instrumented version of Rhino is run with those tests (make sure to build it before using the ant task `buildinstrumented`). Last, all graphs of tests that fail, but are not expected to fail are removed.

1.1.1 Data

None.

1.1.2 Options

1.2 `--mine`

Converter → ParSeMiS → uniq → Scoring

1.2.1 Data

List of packages/classes to consider, when converting to classes/methods.

Option	**Description**
`-fixId=`	ID of the bug to consider (in repository.xml)
`-l=`	Likelihood to include a test
`-engine=`	Which Rhino engine to test against (rhino—rhinoi
`-suffix=`	Suffix to append to the output directory
`-stlg`	Skip test list generation. There must be a file named `sampled-tests.ls` in Rhino‘s test direc

Tabelle 1.1: Options for `--prepare`

1.2.2 Options

1.3 `--class-sampler`

Sample classes to mine. Those within `failing/fix` will all be included, then we fill up randomly.

1.3.1 Data

None.

1.3.2 Options

1.4 `--dot`

Write a serialized graph to a DOT file. All annotations and dummies are included.

1.4.1 Data

`<input file> <output file>`

1.4.2 Options

None.

1.5 `--scoring`

Calulates scores for graph-fragments.

1.5.1 Data

None.

Option	Description	Required	Flag
`-i`	Serialized graph objects to convert	✓	
`<level>`	`-package|-class|-method|-all`	✓	
`[-classList=]`	Valid for -class. LS file of classes to include		
`-writeWeights`	Write weights to LG		✓
`-includeDummies`	Include dummies (foreign packages, classes) into LG		✓
`-includeJre`	Include JRE dummies into LG		✓
`-reincludeDummies`	If dummies (foreign packages, classes) are omitted before, re-include them for entropy ranking		✓
`-reincludeJre`	If dummies are omitted before, re-include those, representing calls to JRE, for entropy ranking		✓
`-skipConstructors`	Omit constructors		✓
`-minFreq=`	Minimum frequency (default=10)		
`-closeGraph`	Use close graph		✓
`-s`	Do not print scoring to stdout		✓
`-wof=`	Write scoring to file. Just pass an ID string here		
`-sc`	Skip the converter		✓
`-sgm=`	Skip the graph-mining step. Pass the fragment file to use.		
`-suffix=`	Suffix to append to produced output.		

Tabelle 1.2: Options for `--mine`

1.5.2 Options

1.6 `--converter`

Converts a repository of serialized graphs (AdjacenceList) to another hierarchy level an prints the corresponding graph DB (as LG).

1.6.1 Data

```
-class [...] [package to consider ]1, 10
-method [...] [class to consider ]1, 10
```

Option	Description	Required	Flag
`-o`	Output file (a list file)	✓	
`-Id=`	BudID	✓	
`-prefix`	Package identifier	✓	
`-n=`	Number of files (.class) to be included	✓	
`-v`	Verbose mode (print selected classes to stdout)		✓

Tabelle 1.3: `bl.tools.ClassSampler`

Option	Description	Required	Flag
`-i`	The fragments file (ParSeMiS)	✓	
`-arff`	Output ARFF file (needed for entropy based scoring in WEKA)	✓	
`-ser`	Path to the serialized graph objects that were used to create the fragments file	✓	
`-reincludeDummies`	If dummies were omitted before (only class level), reinclude for entropy score		✓

Tabelle 1.4: `bl.postprocessor.Scoring`

1.6.2 Options

1.7 `--cleaner`

Deletes all tests we did not expect to fail.

1.7.1 Data

```
<bug id> <path to iBUGS‘s repository.xml> <path to serialized graph objects>
```

1.7.2 Options

None.

1.8 `--copier`

Copies the files mentioned in iBUGS repository.xml as `<testForFix>` from post-fix version to pre-fix version. javascript files in the parent directory of the test are copied as well. Those are usually the included shell files.

Set	Option	Description	Required	Fla
`-package`				
`-class`	`-classList=`	Include only classes in ls file		
`-method`				
`-all`				
All Sets	-i	Directory of serialized graphs	✓	
	-o	Output directory (LG will get sme name)	✓	
	-writeWeights	Write weights into LG		✓
	-includeDummies	Write dummy vertices into LG (only class level)		✓
	-includeJre	Set if JRE calls should be written to LG		✓
	-skipConstructors	Set if constructors should be omitted by the converter		✓

Tabelle 1.5: `bl.tools.Converter`

1.8.1 Data

`<path to Rhino‘s post-fix tests> <bug id> <path to iBUGS repository.xml>`

1.8.2 Options

None.

1.9 `--generator`

Generates a list file that includes tests with the specified likelihood, but includes every test that is in `<iBUGS dir>/output/<fixID>/pre-fix/mozilla/js/tests/failing/fix`.

1.9.1 Data

`<location of the tests> <percent of tests to sample>`

1.9.2 Options

None.