CLI Usage

Overview

Scribble accepts normal Solidity files and has 3 output modes:

1. Inflat
2. mode, all the dependencies of the target Solidity file(s) are bunched up and emitted as one giant blob of Solidity. This is useful for submitting the instrumented code to services like `MythX
3. .
4. Infiles
5. mode, for each filefoo.sol
6. containing contracts with scribble annotations, Scribble emits afoo.instrumented.sol
7. in the same directory and leaves the original unchanged. Note thatscribble
8. may instrument files that are not specified on the command line, but are reachable by a chain of imports.
9. Injson
10. mode, Scribble behaves similarly toflat
11. mode, in that it first flattens all the files necessary to compile the target into one flat Solidity blob. Afterwards, it runssolc
12. on that flat solidity blob and outputs theslightly modified standard JSON output
13. of the used compiler version.
14.

Quick recipes

We provide examples for the most common use cases. For all examples assume we have the following 2 solidity files in the current directory:

Base.sol

```

Copy contract Base { }

```

Foo.sol

```

Copy import "Base.sol"; contract Foo is Base { /// #if_succeeds {:msg "P1"} y == x + 1; function inc(uint x) public pure returns (uint y) { return x+1; } }

```

Emitting a flat instrumented file

If we want to instrumentFoo.sol (and any required imports) and emit it all into a singleFoo.flat.sol we can run:

```

Copy scribbleFoo.sol--output-modeflat--outputFoo.flat.sol

```

Note that both the input and the output can be-- which would make Scribble work withstdin /stdout . The below command usesstdin /stdout to do the same thing as the previous command:

```

Copy catFoo.sol|scribble----output-modeflat--output-->Foo.flat.sol

```

Instrumenting files in-place

Thefiles mode is more involved and is intended for interoperability with various user testing and deployment environments. To instrumentFoo.sol andBase.sol in-place we run:

```

Copy scribbleFoo.sol--output-modefiles
```

```

After running this we will have 2 new files in the directory -Foo.instrumented.sol and__scribble_ReentrancyUtils.sol .Foo.instrumented.sol is the instrumented version ofFoo.sol , and__scribble_ReentrancyUtils.sol contains a helper contract.

Note that the instrumentedFoo.instrumented.sol still imports the originalBase.sol . To actually use the instrumented files in this mode, you have toswap them with the originals, before compiling/testing/deploying. You can do so manually, or automatically with the the--arm and--disarm options to scribble (more detailshere ).

Emitting an instrumented JSON artifact

Emitting an instrumented JSON artifact is almost the same as running in flat-mode:

```
Copy scribbleFoo.sol--output-modejson--outputFoo.flat.json
```

The generatedFoo.flat.json contains the compiled version of the flattened instrumented solidity, along with some extra info. (For more details seethis section ).

Command line options reference

```
Copy Scribble

A Solidity runtime verification tool for property based testing.

Options

-h, --help Print help message -v, --version Print package version -q, --quiet Don't output anything to stderr -i, --input-mode string Input mode. Either 'source' or 'json'. When 'source' is specified, input assumed to be a raw Solidity source code. When 'json' is specified, input assumed to be a Solc output JSON artifact. -m, --output-mode string Output mode. Either 'flat', 'files' or 'json'. When 'flat' is specified, output all instrumented contracts toghether. When 'files' is specified, for every file Foo.sol that contains an annotated contract emit a Foo.sol.instrumetned with the instrumented contract. When 'json' is specified, a JSON object including the original sources, the flattened bytecode, and a map from property IDs to their locations. -k, --keep-instrumented Keep instrumented files after disarming. -o, --output string When the output mode if 'flat' or 'json' specifies where to dump the flattened contracts. Either a path to a file, or '--' for stdout. --utils-output-path string Path to a folder where the ReentrancyUtils.sol contract should be emitted. This option is required with '--mode files' and ignore otherwise. --instrumentation-metadata-file string If specified, output instrumentation metadata (src-to-src maps, property maps) to the specified file. --macro-path string Path to look for a macro definition file. If path is a directory, then it would be recusively scanned for macro files. --path-remapping string Path remapping input for Solc. A semicolon separate list of remappings 'prefix=target' --compiler-version string If given, specifies the exact compiler version to use --compiler-kind string If given, specifies the type of Solidity compiler to use. Supported values are 'wasm' and 'native'. Default value is 'native' (can be configured via environment variable SCRIBBLE_DEFAULT_COMPILER_KIND). --compiler-settings string If given, specifies additional settings to pass to the underlying compiler as a JSON string. (e.g. --compiler-settings '{"optimizer:": {"enabled": true}}'). For more info see https://docs.soliditylang.org/en/latest/using- the-compiler.html#input-description --no-assert If specified execution will not halt when an invariant is violated (only an event will be emitted). --filter-type string If specified instrumentation will happen only for annotations, whose types are matching supplied regular expression. --filter-message string If specified instrumentation will happen only for annotations, whose messages are matching supplied regular expression. --arm When instrumenting files in-place ('--output-mode files') also swap the instrumented files with the original. --disarm Find all dependencies of the target that have been swapped with their instrumented version (all foo.sol s.t. there is a foo.sol.instrumented and foo.sol.original) and swap them back for the original. --debug-events When specified for every failure an additional debug event with values of variables in the failing predicate is emitted. --user-assert-mode string Describe how scribble signals user assertion failures to Myhtril/Harvey. Must be either 'log' or 'mstore'. See wiki for more details. --cov-assertions When specified emit an extra user-assertion for each property whenever its reached. This allows an easy (albeit noisy) way to check which propterties were reached. --solFiles string[] List of solidity files to instrument. Either a list of paths to files, or '--' for stdin.
```

Important caveats

1. --output
2. option doesn't have any effect with--output-mode files
3. .
4. --utils-output-path
5. allows to specify a different directory for the__scribble_ReentrancyUtils.sol
6. file. It only works with--output-mode files
```

7. . It doesn't have any effect with--output-mode flat
8. .
9. If--no-assert
10. is omitted, then there is anassert(false)
11. emitted in the instrumented code every time an invariant is violated. For longer-running fuzzing campaigns, it may be useful to keep running the code even after a property is violated. In those cases only anAssertionFailed(string)
12. event is emitted, which can be caught by tools such as Dili-Faas/MythX/Mythril/Harvey.
13. You will get an error if you passstandard json
14. with missingsources
15. toscribble
16. . Scribble needs the source code to fill in missing information in the AST in certain compiler versions.
17. With the--compiler-kind
18. option you have the choice between using a binary compiler (native
19.) or a WASM based compiler (wasm
20.). If a binary compiler is not available for your environment (e.g. M1 Macs) please use--compiler-kind wasm.
21. When instrumenting in-place (i.e. with the option--output-mode files --arm
22.) scribble will always emit an instrumentation metadata file. By default this is placed in the root of the current project tree. The root is defined as the closest directory with anode_modules/
23. folder in it. If a root folder is not detected, scribble will throw an error requiring you to supply a location for the metadata with the --instrumentation-metadata-file option.
24.

Previous Installation Next Arm and disarm Last updated2 years ago

Was this helpful?