In-place testing with Scribble

A lot of projects come with their own set of tests. Tests are a useful quick check for your properties - if the properties you have in mind don't hold for the user tests, then something might be wrong with them. However in order to build an arbitrary project and run its test suite, Scribble needs to instrument the filesin-place , without changing the file/directory structure, or contracts' external interfaces.

In this tutorial we will learn how to instrument a simple project in-place with scribble, and check whether its tests still pass after instrumentation. A failing test may indicate either a bug in your specifications or a bug in your actual program.

Prerequisites

You should have the following installed:

- git
- node version 12.x
-

Setting-up

You need to first installscribble :

```
Copy npminstall-geth-scribble
```

Next you need to clone the sample project repo:

```
Copy gitclonehttps://github.com/consensys/scribble-getting-started
```

And runnpm install in the directory containing the tutorial:

```
Copy cdscribble-getting-started/in-place-testing npminstall
```

This will installtruffle andganache-cli and any remaining dependencies.

The project

In thescribble-getting-started/in-place-testing directory of the repository you should find a simple Truffle project. Undercontracts/ there are 2 contracts of interest - a base contractManaged that contains some logic for adding and removing administrators and a test contractTest that extendsManaged .

Managed.sol

```
Copy contract Managed { address owner; mapping(address=>bool) admins;

constructor() public { owner = msg.sender; admins[owner] = true; }

modifier OwnerOnly { assert(owner == msg.sender); _; }

modifier AdminOnly { assert(admins[msg.sender]); _; }

function addAdminInternal(address addr) internal { admins[addr] = true; } }
```

Test.sol

```
```

Copy import "./Managed.sol";

contract Test is Managed { function addAdmin(address newAdmin) public { addAdminInternal(newAdmin); } }

```

One property we may want to add toManaged.sol , is that only current administrators may call theaddAdminInternal function. The contractTest violates that property -Test.addAdmin callsaddAdminInternal without first checking if the caller is an admin.

As a first try we can add the following annotation inManaged.sol (extra brownie points if you see something weird already ;)):

```

Copy /// #if_succeeds {:msg "only owner can add admins"} admins[msg.sender]; function addAdminInternal(address addr) internal { admins[addr] = true; }

```

The added annotation is a 'function annotation `. It consists of 3 parts:

1. **if_succeeds**
2. specifies that the predicate should only be checked if theaddAdminInternal
3. function succeeds.
4. {:msg "only owner can add admins"}
5. provides a human-readable message to help remind you what this property is checking. As shorthand you can skip the{:msg }
6. and only specify"only owner can add admins"
7. .
8. admins[msg.sender]
9. is the actual property. Its a valid boolean Solidity expression. In this case we are saying that the value ofadmin[msg.sender]
10. must be true upon successful execution ofaddAdminInternal
11. .
12.

This project has a testtest/tests.js in which a random user tries to add himself as an admin.

```

Copy contract("Test",(accounts)=>{ it("can add admin",()=>{ returnTest.deployed().then(asyncfunction(instance) { awaitinstance.addAdmin(accounts[1],{from:accounts[1]}); }); }) })

```

If we run the test suite without instrumentation this test would actually succeed. We would expect that if we instrumented the contract, and ran the test, the test would fail, showing that the contract doesn't respect the specification.

Instrumenting contracts in-place manually

To instrument the contracts we can run:

```

Copy scribblecontracts/Test.sol--output-modefiles

```

```

Copy Found 1 annotations in 1 different files. contracts/Managed.sol -> contracts/Managed.sol.instrumented

```

The output tells us thatscribble created a.instrumented version forManaged.sol . Indeed we should see the new undercontracts :

```

Copy lscontracts

```

```

Copy Managed.sol Managed.sol.instrumented Migrations.sol __scribble_ReentrancyUtils.sol Test.sol

```

There is a 2nd new file there as well -__scribble_ReentrancyUtils.sol . This is a contract for internal use thatscribble always emits.

Now we want to swap the.instrumented file with its original counterpart and re-run the tests. Since we don't want to loose our original files, lets first copy the originals to the side:

```

Copy cpcontracts/Managed.solcontracts/Managed.sol.original

```

Then copy the instrumented file in-place of the original:

```

Copy cpcontracts/Managed.sol.instrumentedcontracts/Managed.sol

```

Now you should be able to run the test suite and hit the failure. To do so in another shell window you must startganache-cli by running:

```

Copy npmruntestrpc

```

Back in the original shell you can run the tests with:

```

Copy truffletest

```

```

Copy Using network 'development'.

Compiling your contracts...

    Compiling ./contracts/Managed.sol Compiling ./contracts/Test.sol Compiling
    ./contracts/__scribble_ReentrancyUtils.sol Compilation warnings encountered:

/home/dimo/work/consensys/scribble-getting-started/in-place-testing/contracts/__scribble_ReentrancyUtils.sol:2:1: Warning: Source file does not specify required compiler version! Consider adding "pragma solidity ^0.5.16;" contract __scribble_REENTRANCY_UTILS { ^ (Relevant source part starts here and spans across multiple lines).

    Artifacts written to /tmp/test-202066-9871-gaivy6.q7obe Compiled successfully using: - solc:
    0.5.16+commit.9c3226ce.Emscripten.clang

Contract: Test ✓ can add admin (77ms) 1 passing (104ms)

```

But wait! why did that test succeed? Shouldn't it have failed since the sender wasn't an administrator?

The key here is when theadmin[msg.sender] predicate was evaluated - it was evaluated after the function had successfully executed, at which point the sender had already added themselves to theadmin map. To fix this, we want to talk about the value ofadmin[msg.sender] before the function started executing. We can do this using theold() operator, which specifies thatscribble should compute the value of a given expressionbefore the function is called.
```

So lets fix this. First lets move the original annotated file back:

```
Copy cpcontracts/Managed.sol.originalcontracts/Managed.sol
```

Now lets change the annotation to look like this:

```
Copy /// #if_succeeds {:msg "only owner can add admins"} old(admins[msg.sender]); function addAdminInternal(address addr) internal { admins[addr] = true; }
```

Now if we repeated the whole process of instrumenting, swapping the files, and running tests we should hit an error. But that process is so laborious! There must be a better way! And there is - introducing--arm and--disarm .

Scribble --arm and --disarm

Copying files back and forth, keeping track of.sol.instrumented and.sol.original files is annoying. Soscribble provides you with two options that take care of this for you.

If you add the--arm option when instrumenting, thenscribble will automatically create.sol.original copies of the original files, and swap the instrumented files in-place, so you can start testing immediately. So now that we have fixed our annotation, lets use--arm and--disarm .

First we will instrument the contracts and swap them in-place with a single command:

```
Copy scribblecontracts/Test.sol--output-modefiles--arm
```

```
Copy Found 1 annotations in 1 different files. contracts/Managed.sol -> contracts/Managed.sol.instrumented Copying contracts/Managed.sol to contracts/Managed.sol.original Copying contracts/Managed.sol.instrumented to contracts/Managed.sol
```

Note that the new output shows which files were copied where. At this point if we run the tests, we can verify that the fixed annotation fails:

```
Copy truffletest
```

```
Copy ... 1) Contract: Test can add admin: Error: Returned error: VM Exception while processing transaction: invalid opcode at /home/dimo/work/consensys/scribble-getting-started/in-place-testing/test/test.js:7:32 at process._tickCallback (internal/process/next_tick.js:68:7)
```

And afterwards we can revert the workspace back by using the--disarm option:

```
Copy scribblecontracts/Test.sol--disarm
```

```
Copy Moving contracts/Managed.sol.original to contracts/Managed.sol Removing contracts/Managed.sol.instrumented
```

Removing /home/dimo/work/consensys/tmp/test_pass/scribble-getting-started/in-place-testing/contracts/__scribble_ReentrancyUtils.sol Removing /home/dimo/work/consensys/tmp/test_pass/scribble-getting-started/in-place-testing/instrumentation.scribble.json

```

Note that the additional.original and.instrumented files, as well as other auxilary files emitted by scribble are all removed:

```

Copy lscontracts/

```

```

Copy Managed.sol Migrations.sol Test.sol

```

If you want to keep the.instrumented files for debugging purposes, add the--keep-instrumented option when disarming.

Testing with and without asserts

When we ran the instrumented tests, we hit an assertion and the test fails. However if we had multiple properties, we wouldn't know WHICH property fails. It would be nice to get a message telling us which property failed. To do so, lets re-instrument the code with the--no-assert option:

```

Copy scribblecontracts/Test.sol--output-modefiles--arm--no-assert

```

```

Copy Found 1 annotations in 1 different files. contracts/Managed.sol -> contracts/Managed.sol.instrumented Copying contracts/Managed.sol to contracts/Managed.sol.original Copying contracts/Managed.sol.instrumented to contracts/Managed.sol

```

The--no-assert option tells scribble to not emit an explicitassert(false) on failure, and instead just emit an event. As a result, the tests may not fail anymore, but if you run them with--show-events you will see all the failing properties:

```

Copy truffletest--show-events

```

```

Copy Using network 'development'.

Compiling your contracts...

> Compiling ./contracts/Managed.sol Compiling ./contracts/Test.sol Compiling ./contracts/__scribble_ReentrancyUtils.sol ...

Contract: Test ✓ can add admin (72ms)

Events emitted during test:

Managed.AssertionFailed(message: '0: only owner can add admins' (type: string))

---

1 passing (177ms)

```
```

Now we see theAssertionFailed event. Its text contains the human-readable label from the original annotation.

Previous Property Checking with Scribble and Mythril Last updated2 years ago