Assertion Checking Modes

There are several options for what Scribble can do when an assertion fails. By default Scribble will emit anassert (false) and an additional log statementemit AssertionFailed("..") . For example, given this annotated code:

```

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

```

The instrumented code would look like this:

```

Copy contract Foo { event AssertionFailed(string message);

function inc(uint x) public returns (uint y) { y = _original_Foo_inc(x); if ((!((y == (x + 1))))) { emit AssertionFailed("0: P1"); assert(false); } }

function _original_Foo_inc(uint x) private pure returns (uint y) { return (x + 1); } }

```

Theemit AssertionFailed("0: P1") statement on line 7 is used to signal to tools such asMythril andHarvey that a user property violation was found . It can be also useful for the end user to identify which property failed in normal test runs by looking at the emitted events.

No assertions

Sometimes for long running tests and fuzzing campaigns we don't want to interrupt the tests for every violation found. In those cases you can preventscribble from emittingassert(false) with the--no-assert option.

User assertion failures without event logging

In some cases signaling a user property violation using event logging may be inconvenient, since the assertion failure check may occur in aview function. In such cases adding anemit statement changes the mutability of the instrumented function, which can change the interfaces of the compiled contracts and may break tests. For this reasonscribble (and Mythril and Harvey) support an alternative mode for signaling custom property violations using memory stores with specially crafted bit-strings. You can select this mode with the--user-assert-mode mstore command line option. If you were to instrument our earlier exmaple in this mode the instrumented code would look as such:

```

Copy contract Foo { struct vars0 { uint256**mstore_scratch**; }

function inc(uint x) public pure returns (uint y) { vars0 memory _v; y = _original_Foo_inc(x); { _v**mstore_scratch** = 0xcafecafecafecafecafecafecafecafecafecafecafecafecafecafe1000; if ((!((y == (x + 1))))) { _v.**mstore_scratch** = 0xcafecafecafecafecafecafecafecafecafecafecafecafecafecafe0000; assert(false); } } }

function _original_Foo_inc(uint x) private pure returns (uint y) { return (x + 1); } }

```

In this mode the assertion failure is signaled by the store to_v**mstore_scratch** on line 12. Tools such as Myhtril and Harvey can watch for assignments to memory with the0xcafe pattern appended with the id of the failing assertion.

Previous Arm and disarm Next Instrumented Code Last updated2 years ago

On this page * No assertions * User assertion failures without event logging

Was this helpful?