Formal Verification with The Certora Prover

Michael George

Product Director, Certora

DeFi Security Summit

Stanford, August 2022

Formal Verification with the Certora Prover

The Certora Prover is a tool for finding bugs in smart contracts

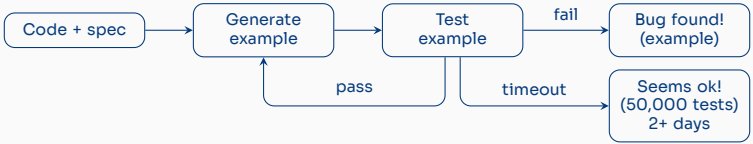
- Developers specify the intended behavior of the contract
  - Prover uses specs written in Certora Verification Language (CVL)
  - Specs are like unit tests ...but infinitely more powerful
- The Prover checks that the contracts obey those properties
  - in all circumstances! (every possible storage, every possible input)

The Prover relies on results of decades of research in formal verification

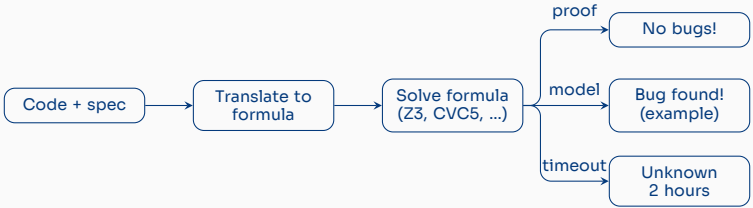
- Both academic and industrial

Formal Verification vs. Fuzzing

Fuzzing:



Formal verification (Certora Prover):



certora

Certora Verficiation Language (CVL) Features:

- Solidity-like syntax
  - Specs can call methods, assign variables, define functions, ...
  - Use `require` to write preconditions, `assert` to describe behavior
- Reason about arbitrary values
  - Prover considers every possible combination of values for undefined variables
  - Prover considers every possible combination of values of storage variables
- Syntax for calling an arbitrary method
  - e.g. if any method increases any user's allowance, the user was the sender
- Access internal contract state
  - e.g. everytime `_balances[a]` changes, update `sum_balances`
- Explicit syntax for writing state invariants
  - ```
    invariant totalSupplyBoundsBalances(address a)
        balanceOf(a) <= totalSupply()
    ```
- Reason about reverting paths
  - e.g. in emergency mode, `withdraw` never reverts unless ...
- Rewind storage to a previous state
  - e.g. Rerunning with more permissions doesn't cause revert

certora
    ```

Example / demo

```
/// Transfer must move `amount` tokens from
/// the caller's account to `recipient`.
rule transferSpec {
    address sender; address recip; uint amount;

    env e;
    require e.msg.sender == sender;

    mathint balance_sender_before = balanceOf(sender);
    mathint balance_recip_before  = balanceOf(recip);

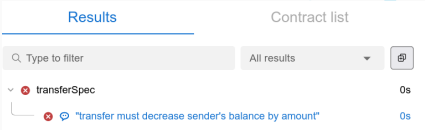
    transfer(e, recip, amount);

    mathint balance_sender_after = balanceOf(sender);
    mathint balance_recip_after  = balanceOf(recip);



    assert balance_sender_after == balance_sender_before - amount,
        "transfer must decrease sender's balance by amount";

    assert balance_recip_after  == balance_recip_before  + amount,
        "transfer must increase recipient's balance by amount";
}
```



certora

Example / demo

```
/// Transfer must move `amount` tokens from
/// the caller's account to `recipient`.
rule transferSpec {
    address sender; address recip; uint amount;

    env e;
    require e.msg.sender == sender;

    mathint balance_sender_before = balanceOf(sender);
    mathint balance_recip_before  = balanceOf(recip);

    transfer(e, recip, amount);

    mathint balance_sender_after = balanceOf(sender);
    mathint balance_recip_after  = balanceOf(recip);



    assert balance_sender_after == balance_sender_before - amount,
        "transfer must decrease sender's balance by amount";

    assert balance_recip_after  == balance_recip_before  + amount,
        "transfer must increase recipient's balance by amount";
}
```

Variables | Call resolution

Local Variables

balance_sender_before	2
recip	0xffff
balance_recip_before	2
amount	2
sender	0xffff
balance_sender_after	2
e.msg.sender	0xffff
e.block.coinbase	0x401
e.msg.value	0
e.msg.address	3
balance_recip_after	2

certora

Example / demo

```
/// Transfer must move `amount` tokens from
/// the caller's account to `recipient`.
rule transferSpec {
    address sender; address recip; uint amount;

    env e;
    require e.msg.sender == sender;

    mathint balance_sender_before = balanceOf(sender);
    mathint balance_recip_before  = balanceOf(recip);

    transfer(e, recip, amount);

    mathint balance_sender_after = balanceOf(sender);
    mathint balance_recip_after  = balanceOf(recip);

    require sender != recip;

    assert balance_sender_after == balance_sender_before - amount,
        "transfer must decrease sender's balance by amount";

    assert balance_recip_after  == balance_recip_before  + amount,
        "transfer must increase recipient's balance by amount";
}
```

Job ID	Status
ccb265240dc4db801fad	SUCCEEDED
Start 08/23/2022 08:40:29	**Duration** 00:00:24

Results | Contract list

Type to filter | All results

transferSpec 0s

certora

Verification is only as good as the specification

We're currently developing tools to find bugs in specifications:

- Vacuity checking, tautology checking
  - Sanity checks to flag rules that couldn't possibly catch bugs
- Bug injection
  - Verifying versions with known bugs against the spec
  - Specs with good coverage should catch them!
- Mututation testing
  - Automatically change the code in ways that probably introduce bugs
  - e.g. remove method decorators, change `require` statements, ...
  - Specs with good coverage should catch them!

Thank you!

Questions?

https://demo.certora.com

certora