Ghosts

Michael George

Stanford, August 2022

A (much) stronger invariant

Before the break:

- Tried to show that each user balance is at most the total supply

Now:

- We'll show that the total supply is the sum of all user balances

$$\texttt{totalSupply()} = \sum_{\texttt{a} \in \texttt{address}} \texttt{balanceOf(a)}$$

- It's hard to track infinite sum
  - we'll track changes to balances instead

Hooks

A CVL hook allows running CVL code when the contract updates storage

- Syntax:

```
hook Sstore <pattern> <new variable> (<old variable>) STORAGE {
    <body>
}
```

- Example:

```
hook Sstore _balances[KEY address a] uint new_value (uint old_value) STORAGE {
    ...
}
```

- Pattern is a field followed by any number of:
  - array lookups (using `[INDEX <type> <name>]`),
  - mapping lookups (using `[KEY <type> <name>]`),
  - struct field lookups (using `.field`)

- Hook can update our tracked sum of balances

Ghosts

A ghost variable is an additional variable that doesn't exist in the contract

- Primarily useful for keeping track of changes from hooks

Example:

```
ghost mathint sum_of_balances;
```

You can also declare ghost mappings:

```
ghost mapping(address => mapping(address => uint256)) balances_by_token;
```

Prover considers every possible value of ghost (just like storage)

Putting ghost and hook together

Example (results link):

```
ghost mathint sum_of_balances {
    init_state axiom sum_of_balances == 0;
}

hook Sstore _balances[KEY address a] uint new_value (uint old_value) STORAGE {
    // when balance changes, update ghost
    sum_of_balances = sum_of_balances + new_value - old_value;
}

invariant totalSupplyIsSumOfBalances()
    totalSupply() == sum_of_balances
```

Rule passes on preservation but fails on initialization

- Prover chooses non-zero initial value for the ghost

Initial state axiom tells prover to make assumptions about the intial value of the ghost (before the constructor)

Exercise

- Create a ghost to track the number of changes to users' balances
- Use it to prove that no method changes more than two balances