Formal Verification

Proves that specified invariant will hold under every single possible circumstance.

Lets say $f(x, y, z)$ is function

where $x > y$

$y > z$

$x > z$

$f(x, y, z) = (x > y) \wedge (y > z) \wedge (x > z)$

which can be translated to Solidity

```
contract SMTSChecker1 {
    function verify(uint256 x, uint256 y, uint256 z) external pure  {
        assert(x > y);
        assert(y > z);
        assert(x > z);
    }
}
```

and this code snippet can be formally

verified by using SMT Solvers

SMT Solvers

- K Lang
- SmtLib ⟶ Z3 (Approver)
- SMT Checker (Solc builtin)

SMT Solver Mechanism

Checks unsatisfiability of the negation of boolean formula. i.e.

$$f'(x,y,z) = \neg((x>y) \wedge (y>z) \wedge (x>z))$$

$$\Rightarrow f'(x,y,z) = (x \leq y) \vee (y \leq z) \vee (x \leq z)$$

if negation is satisfied then f() won't hold under all inputs.

Making Function verifiable

```
contract SMTSChecker1 {
    function verify(uint256 x, uint256 y, uint256 z) external pure {
        require(y < x);
        require(z < y);
```

this will more sure

asserts hold.

```
        assert(x > y);
        assert(y > z);
        assert(x > z);
    }
}
```