A Formal Verification Tool for Ethereum VM Bytecode

Daejun Park Yi Zhang Manasvi Saxena
Philip Daian Grigore Rosu

Nov 7, 2018 @ FSE'18

ILLINOIS

runtime verification

Smart contracts

- Programs that run on blockchain
- Usually written in a high-level language
  - Solidity (JavaScript-like), Vyper (Python-like), ...
- Compiled down to VM bytecode
  - EVM (Ethereum VM), IELE (LLVM-like VM), ...

    our target
  - Runs on VM of blockchain nodes

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] =+ value;
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

🤔

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] =+ value;
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

'=+'   vs   '+='

* ETHNews.com, "Ether.Camp's HKG Token Has A Bug And Needs To Be Reissued"

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] = (+value);
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

* ETHNews.com, “Ether.Camp’s HKG Token Has A Bug And Needs To Be Reissued”

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] = value;
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

* ETHNews.com, “Ether.Camp’s HKG Token Has A Bug And Needs To Be Reissued”

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] += value;
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

🤔

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] += value;
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

arithmetic overflow

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] = SafeMath.add(balances[to], value);
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

will throw if overflow

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] = SafeMath.add(balances[to], value);
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

🤔

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] = SafeMath.add(balances[to], value);
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

self-transfer may fail

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {

        balances[from] -= value;
        balances[to] = SafeMath.add(balances[to], value);
        return true;
    } else {
        return false;
    }
}
```

more robust

Why bytecode?

```
interface Token {
  function transfer() returns (bool);
}

contract Wallet {
  function transfer(address token) {
    return Token(token).transfer();
  }
}
```

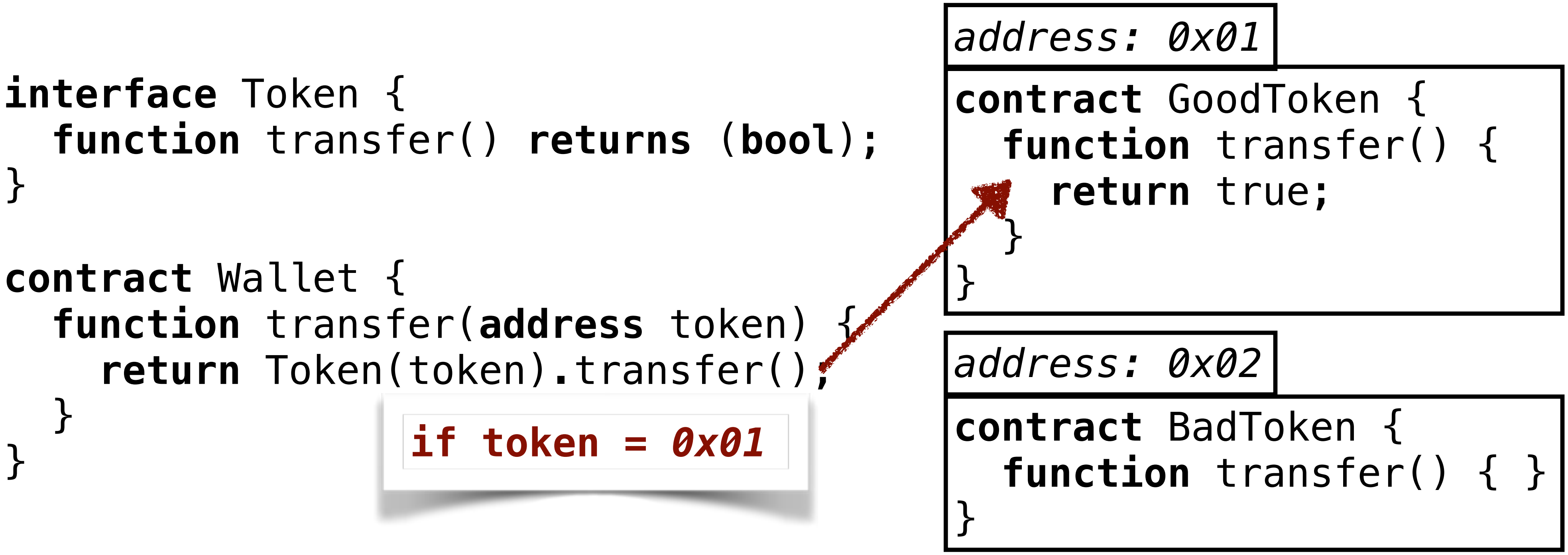
address: 0x01

```
contract GoodToken {
  function transfer() {
    return true;
  }
}
```

address: 0x02

```
contract BadToken {
  function transfer() { }
}
```

Why bytecode?



```
interface Token {
  function transfer() returns (bool);
}

contract Wallet {
  function transfer(address token) {
    return Token(token).transfer();
  }
}
```

```
if token = 0x01
```

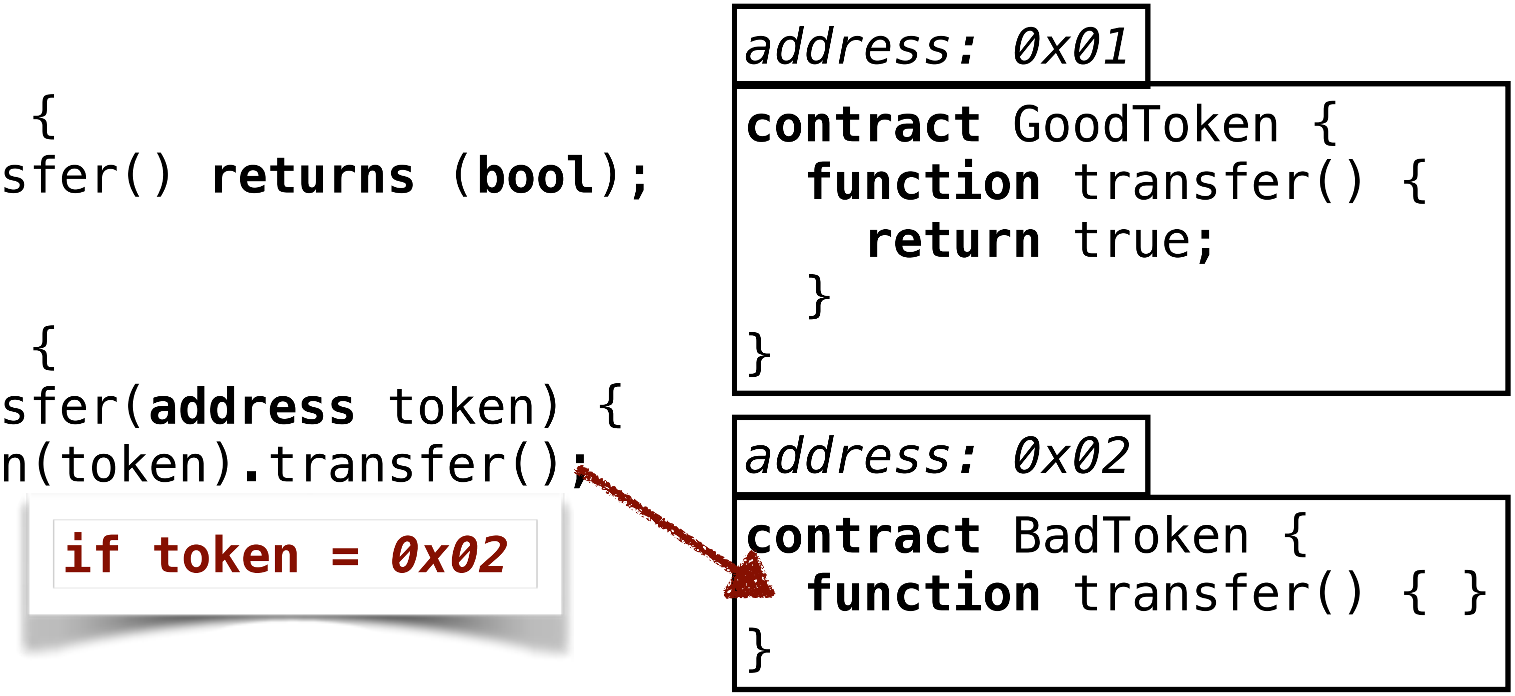
```
address: 0x01
contract GoodToken {
  function transfer() {
    return true;
  }
}
```

```
address: 0x02
contract BadToken {
  function transfer() { }
}
```

Why bytecode?

```
interface Token {
  function transfer() returns (bool);
}

contract Wallet {
  function transfer(address token) {
    return Token(token).transfer();
  }
}
```



```
if token = 0x02
```

```
address: 0x01
contract GoodToken {
  function transfer() {
    return true;
  }
}
```

```
address: 0x02
contract BadToken {
  function transfer() { }
}
```

Why bytecode?

```
interface Token {
  function transfer() returns (bool);
}

contract Wallet {
  function transfer(address token) {
    return Token(token).transfer();
  }
}
```

🤔

`if token = 0x02`

address: 0x01

```
contract GoodToken {
  function transfer() {
    return true;
  }
}
```

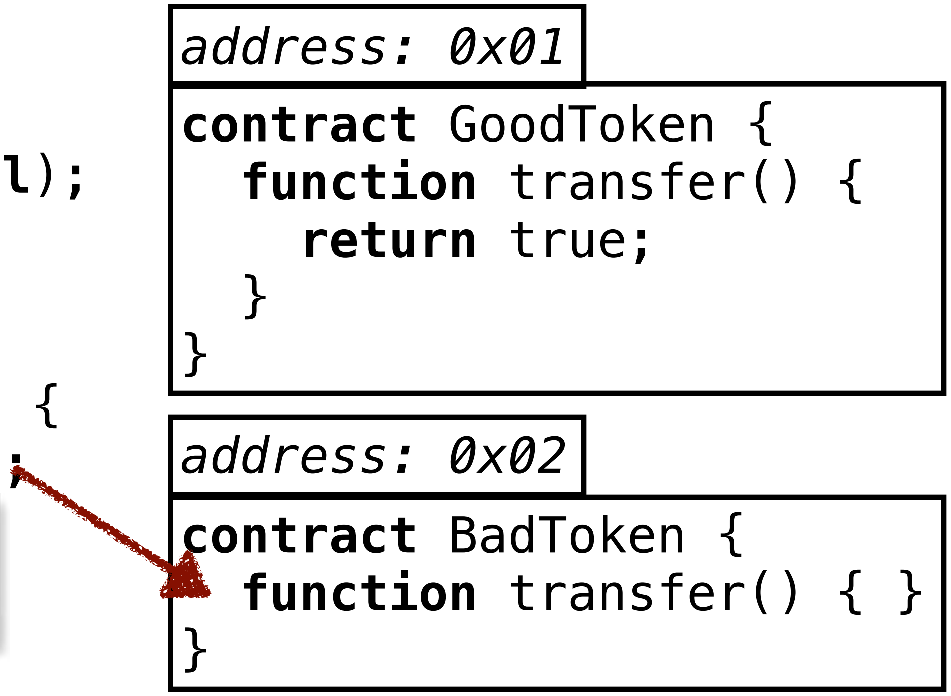
address: 0x02

```
contract BadToken {
  function transfer() { }
}
```

Why bytecode?

```
interface Token {
  function transfer() returns (bool);
}

contract Wallet {
  function transfer(address token) {
    return Token(token).transfer();
  }
}
```



if token = 0x02

address: 0x01

```
contract GoodToken {
  function transfer() {
    return true;
  }
}
```

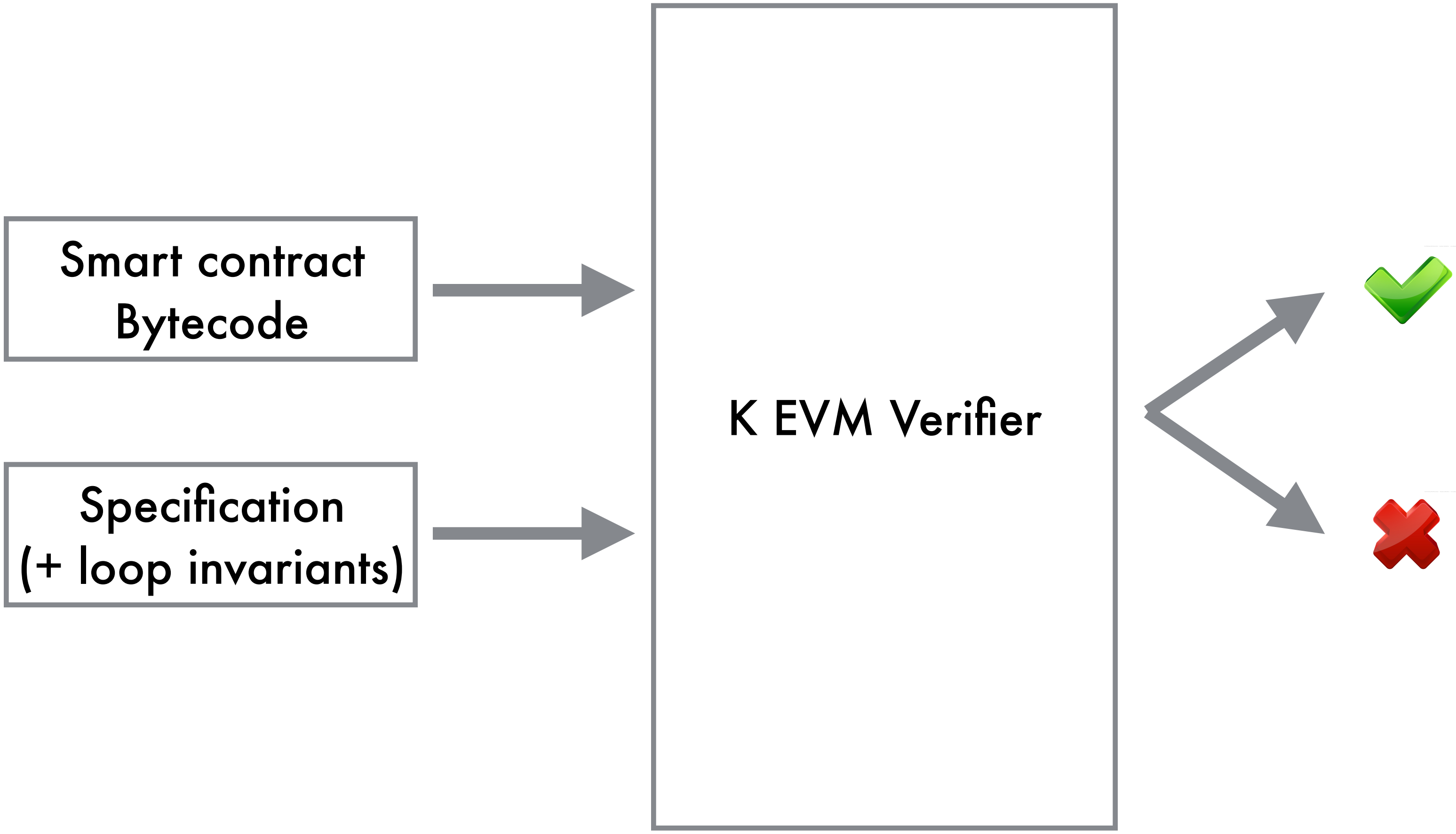
address: 0x02

```
contract BadToken {
  function transfer() { }
}
```

- *Return true* in Solidity 0.4.21 or earlier
- *Revert* in Solidity 0.4.22 or later (latest: 0.4.25)

* Lukas Cremer, “Missing return value bug—At least 130 tokens affected”

K EVM Verifier

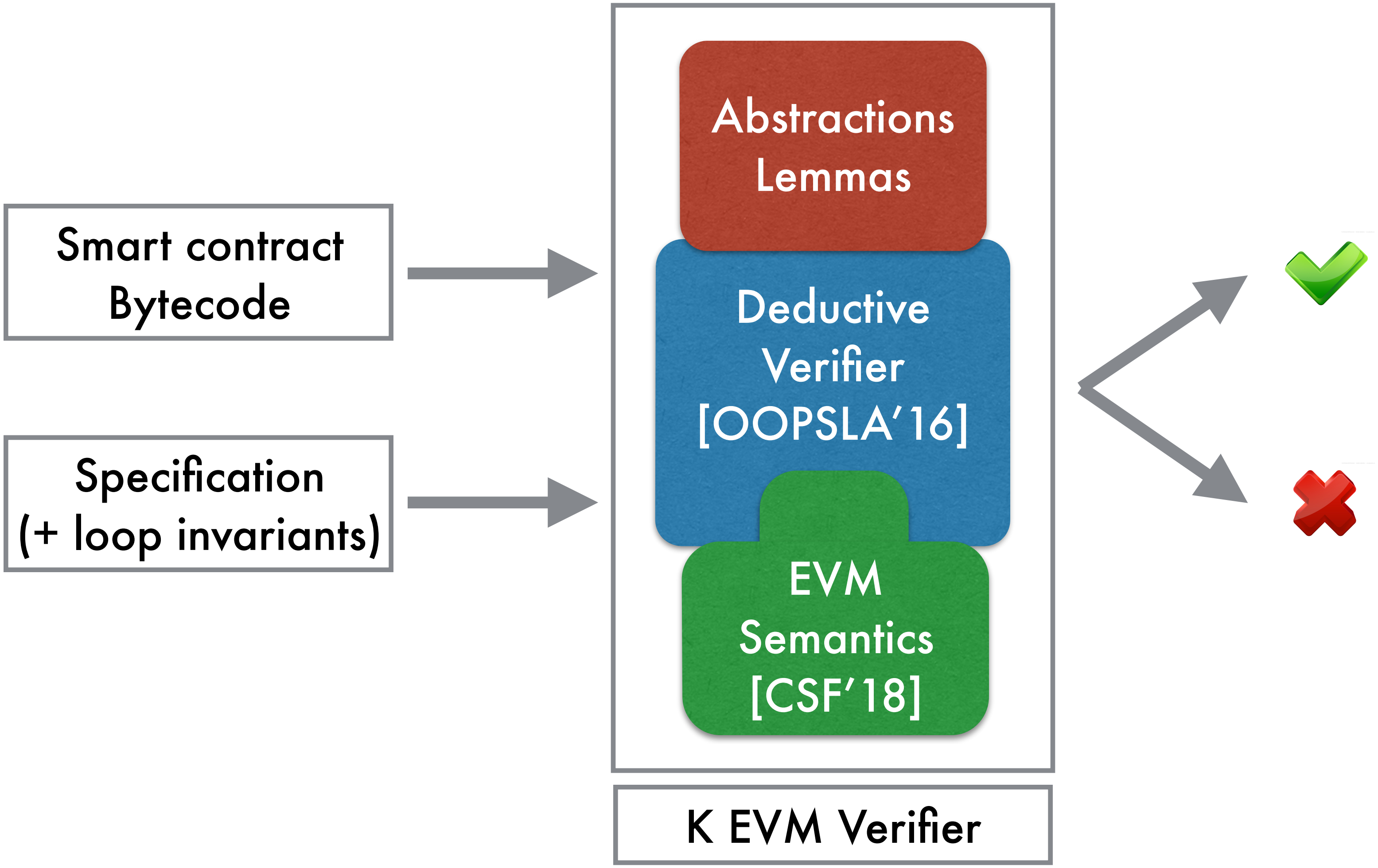


Smart contract Bytecode

Specification (+ loop invariants)

K EVM Verifier

K EVM Verifier



Smart contract Bytecode

Specification (+ loop invariants)

Abstractions Lemmas

Deductive Verifier [OOPSLA'16]

EVM Semantics [CSF'18]

K EVM Verifier

✓

✗

Specification example

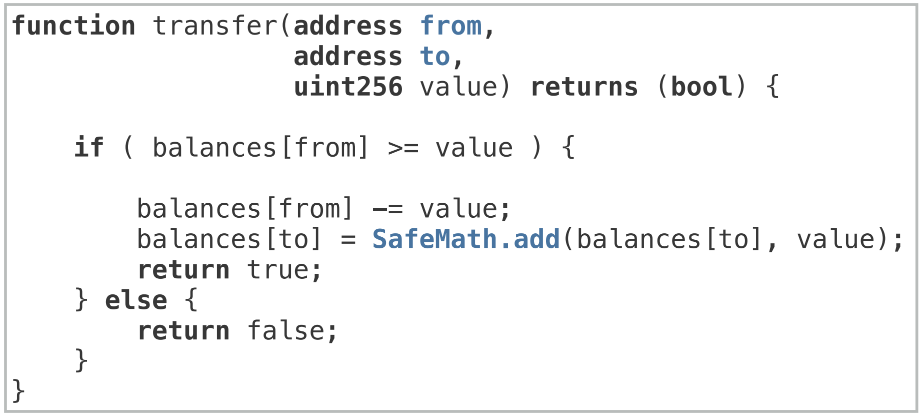
```
[transfer-success]

callData:
  #abiCallData("transfer",
      #address(FROM), #address(TO), #uint256(VALUE))

storage:
  #(BALANCES[FROM]) ⟼ (BAL_FROM ⟹ BAL_FROM - VALUE)
  #(BALANCES[TO]  ) ⟼ (BAL_TO   ⟹ BAL_TO   + VALUE)

requires:
  FROM ≠ TO
  VALUE ≤ BAL_FROM
  BAL_TO + VALUE < (2 ^ 256)
                               true
output:                         ↓
  _ ⟹ #asByteArray(1, 32)

statusCode:
  _ ⟹ EVMC_SUCCESS
```



```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {

        balances[from] -= value;
        balances[to] = SafeMath.add(balances[to], value);
        return true;
    } else {
        return false;
    }
}
```

Verified smart contracts*

- High-profile ERC20 token contracts
- Ethereum Casper FFG (Hybrid PoW/PoS)
- Gnosis MultiSigWallet (ongoing)
- DappHub MakerDAO (by DappHub)
- Uniswap (decentralized exchange)
- Bihu (KEY token operation)

* https://github.com/runtimeverification/verified-smart-contracts

Challenges for EVM bytecode verification

- Byte-twiddling operations
  - Non-linear integer arithmetic (e.g., modulo reduction)
- Arithmetic overflow detection
- Gas limit
  - Variable gas cost depending on contexts
- Hash collision

Byte-twiddling operations

Given:

$$x[n] \overset{\texttt{def}}{=} (x/256^n) \bmod 256$$

$$merge(x[i..j]) \overset{\texttt{def}}{=} merge(x[i..j+1]) \;\underline{\texttt{*}}\; 256 \;\underline{\texttt{+}}\; x[j] \quad \text{when } i > j$$

$$merge(x[i..i]) \overset{\texttt{def}}{=} x[i]$$

Prove:

$$\text{``}x = merge(x[31..0])\text{''.}$$

Abstractions

```
syntax Int ::= nthByte(Int, Int, Int) [function]


rule merge(nthByte(V, 0, N) ... nthByte(V, N−1, N))
  ⟹ V
  requires 0 ≤ V <  2 ^ (N ∗ 8)
       and 1 ≤ N ≤ 32
```

Challenges for EVM bytecode verification

- Byte-twiddling operations
  - Non-linear integer arithmetic (e.g., modulo reduction)
- Arithmetic overflow detection
- Gas limit
  - Variable gas cost depending on contexts
- Hash collision

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {

        balances[from] -= value;
        balances[to] = SafeMath.add(balances[to], value);
        return true;
    } else {
        return false;
    }
}
```

Why bytecode?

```
interface Token {
  function transfer() returns (bool);
}

contract Wallet {
  function transfer(address token) {
    return Token(token).transfer();
  }
}
```

if token = 0x02

address: 0x01

```
contract GoodToken {
  function transfer() {
    return true;
  }
}
```

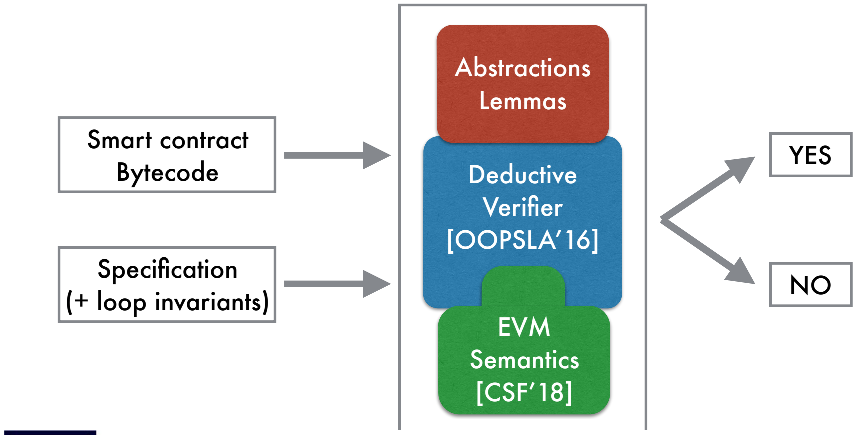
address: 0x02

```
contract BadToken {
  function transfer() { }
}
```

- *Return true* in Solidity 0.4.21 or earlier
- *Revert* in Solidity 0.4.22 or later

https://github.com/runtimeverification/verified-smart-contracts

K EVM Verifier



Specification example

```
[transfer-success]

callData:
  #abiCallData("transfer", #address(TO), #uint256(VALUE))

storage:
  #(BALANCES[FROM]) ↦ (BAL_FROM ⟹ BAL_FROM - VALUE)
  #(BALANCES[TO]  ) ↦ (BAL_TO   ⟹ BAL_TO   + VALUE)

requires:
  FROM ≠ TO
  VALUE ≤ BAL_FROM
  BAL_TO + VALUE < (2 ^ 256)

statusCode:
  _ ⟹ EVMC_SUCCESS
                              true
                               ↓
output:
  _ ⟹ #asByteArray(1, 32
```

Backup

Overflow bug exploit

```
function batchTransfer(address[] receivers, uint256 value)
    public whenNotPaused returns (bool) {

                                overflow
  uint cnt = receivers.length;  ↓
  uint256 amount = uint256(cnt) * value;
  require(cnt > 0 && cnt <= 20);
  require(value > 0 && balances[msg.sender] >= amount);

  balances[msg.sender] = balances[msg.sender].sub(amount);

  for (uint i = 0; i < cnt; i++) {
    balances[receivers[i]] = balances[receivers[i]].add(value);
    Transfer(msg.sender, receivers[i], value);
  }

  return true;
}
```

missed by both Oyente and Securify at that time

* https://twitter.com/vietlq/status/989266840315727872

* https://twitter.com/vietlq/status/989348032046157824