Presented at FMBC 2020

Formal Specification and Verification of Solidity Contracts with Events

Ákos Hajdu[1], Dejan Jovanović[2], Gabriela Ciocarlie[2]

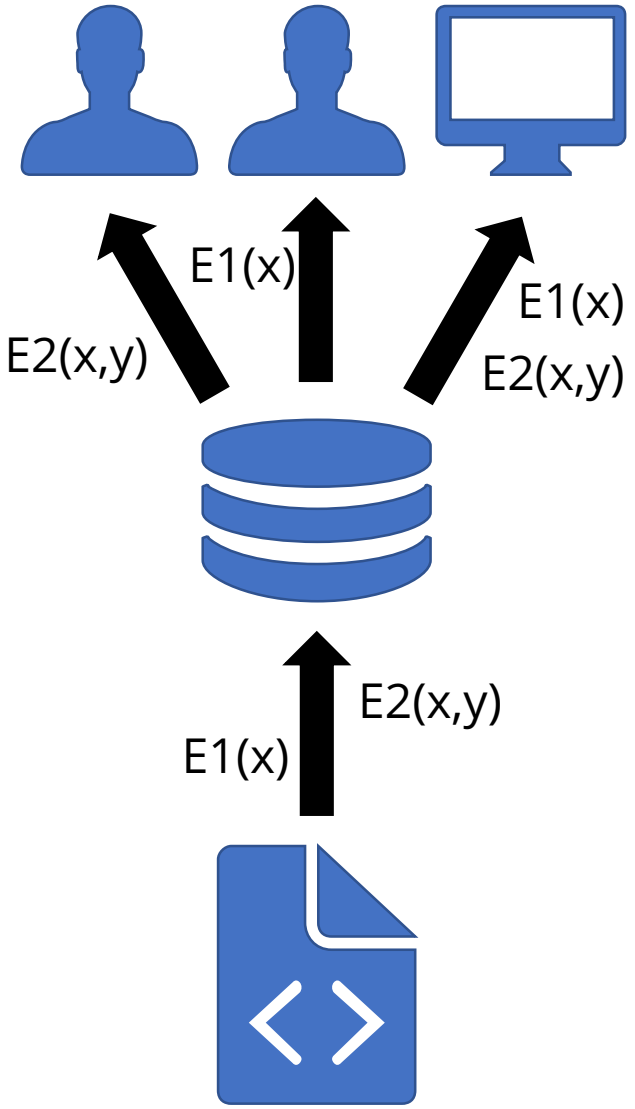
[1]*Budapest University of Technology and Economics*
[2]*SRI International*

Solidity Smart Contracts and Events

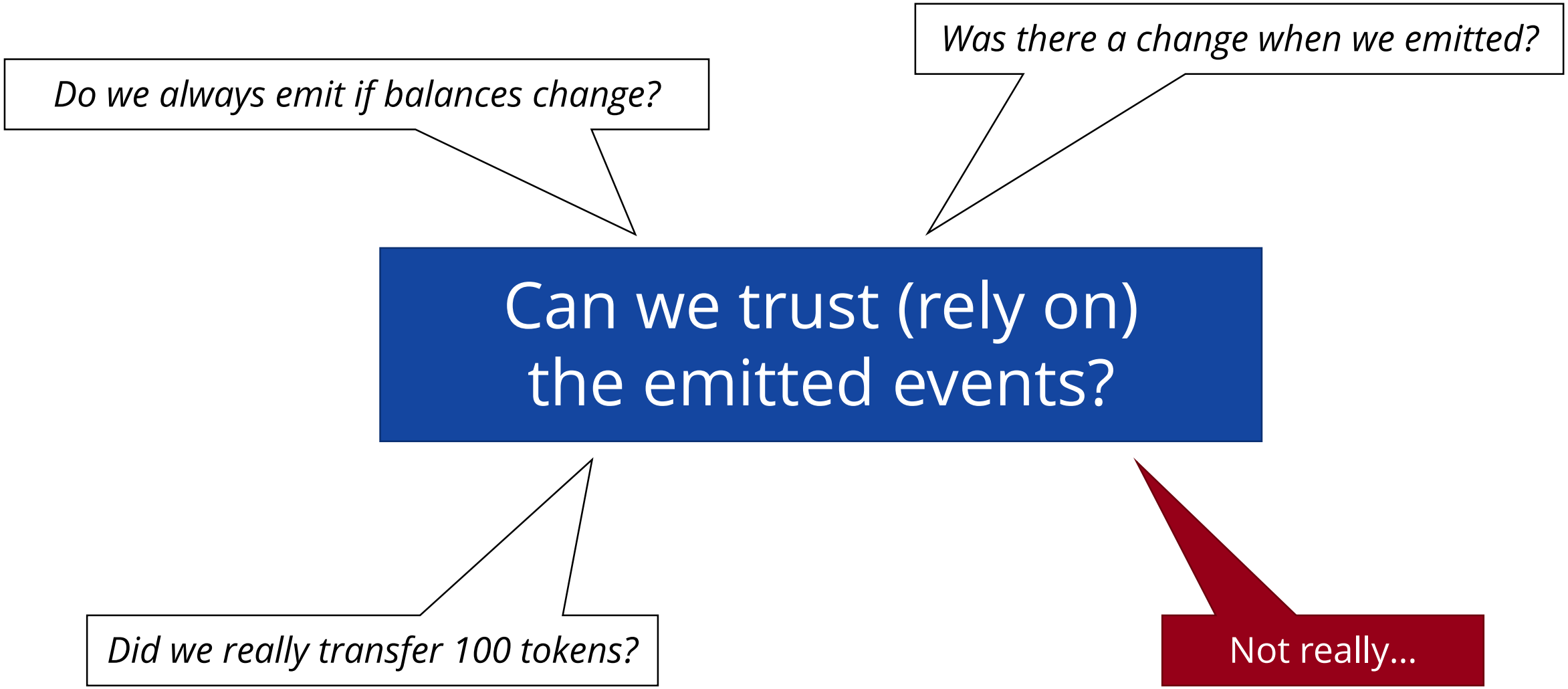
```
contract Token {
  mapping(address=>uint) balances;

  event transferred(address from, address to, uint amount);

  function transfer(address to, uint amount) public {
    require(balances[msg.sender] >= amount && msg.sender != to);
    balances[msg.sender] -= amount;
    balances[to] += amount;
    emit transferred(msg.sender, to, amount);
  }
}
```

Solidity Events

- Stored in blockchain logs
- Contract communicates with user
  - Important state changes
- Abstract view of execution
  - Relevant aspect to each user



Motivation


Was there a change when we emitted?
Do we always emit if balances change?
Can we trust (rely on) the emitted events?
Did we really transfer 100 tokens?
Not really...

Formal Specification of Events

```
contract Token {
  mapping(address=>uint) balances;

  /// @notice tracks-changes-in balances
  /// @notice precondition  balances[from] >= amount
  /// @notice postcondition balances[from] == before(balances[from]) - amount
  /// @notice postcondition balances[to] == before(balances[to]) + amount
  event transferred(address from, address to, uint amount);

  /// @notice emits transferred
  function transfer(address to, uint amount) public {
    require(balances[msg.sender] >= amount && msg.sender != to);
    balances[msg.sender] -= amount;
    balances[to] += amount;
    emit transferred(msg.sender, to, amount);
  }
}
```

Emit iff there is a change

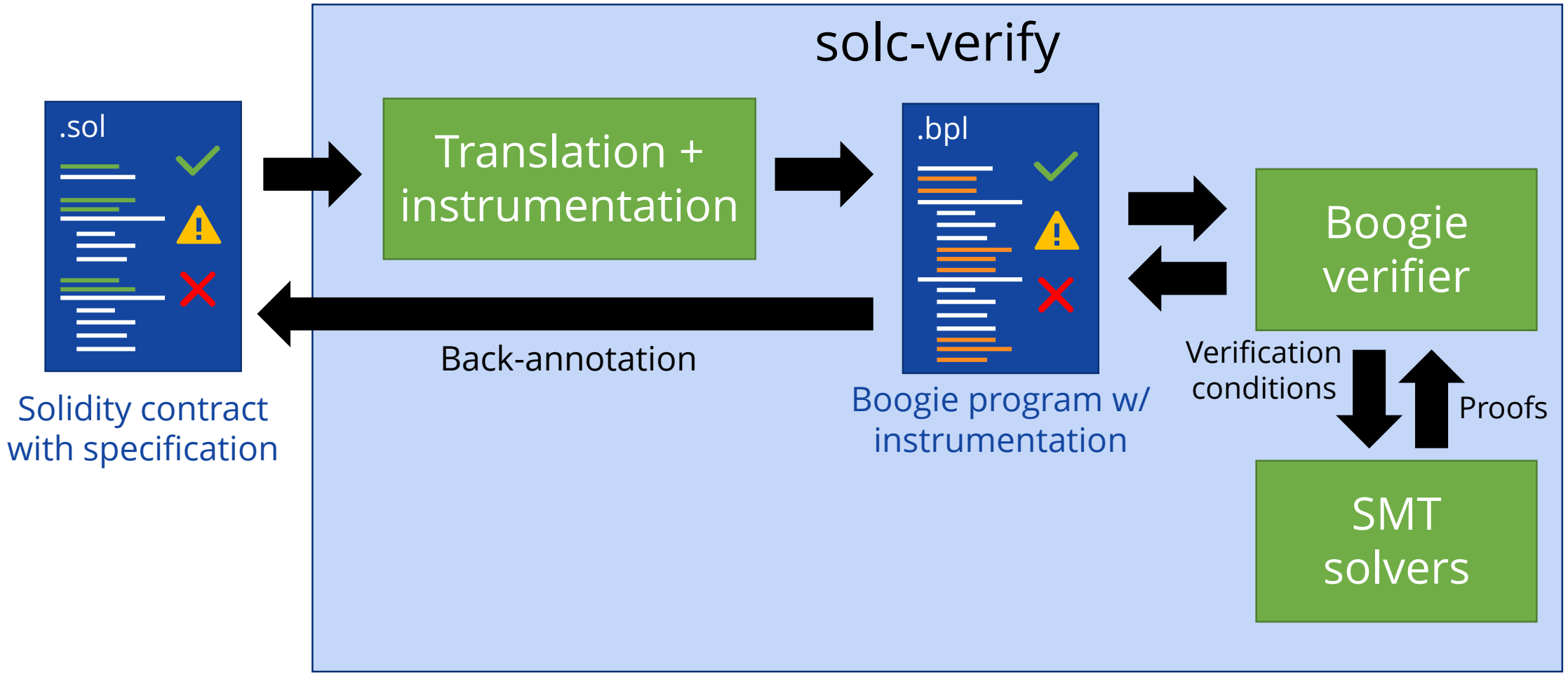
Condition(s) before the change

Condition(s) after the change

Can possibly emit

github.com/SRI-CSL/solidity

Formal Verification of Events



Example

```
contract Token {
  mapping(address=>uint) balances;

  /// @notice tracks-changes-in balances
  /// @notice precondition  balances[from] >= amount
  /// @notice postcondition balances[from] == before(balances[from]) - amount
  /// @notice postcondition  balances[to] == before(balances[to]) + amount
  event transferred(address from, address to, uint amount);

  /// @notice emits transferred
  function transfer(address to, uint amount) public {
    require(balances[msg.sender] >= amount && msg.sender != to);
    balances[msg.sender] -= amount;
    balances[to] += amount;
    //emit transferred(msg.sender, to, amount);
  }
}
```

```
$ solc-verify.py SimpleToken.sol
Token::transfer: ERROR
 - SimpleToken.sol:10:5: Function can end without triggering event
Errors were found by the verifier.
```

Example

```
contract Token {
  mapping(address=>uint) balances;

  /// @notice tracks-changes-in balances
  /// @notice precondition  balances[from] >= amount
  /// @notice postcondition balances[from] == before(balances[from]) - amount
  /// @notice postcondition  balances[to] == before(balances[to]) + amount
  event transferred(address from, address to, uint amount);

  /// @notice emits transferred
  function transfer(address to, uint amount) public {
    require(balances[msg.sender] >= amount && msg.sender != to);
    balances[msg.sender] -= amount;
    balances[to] += amount;
    emit transferred(msg.sender, msg.sender, amount);
  }
}
```

```
$ solc-verify.py SimpleToken.sol
Token::transfer: ERROR
 - SimpleToken.sol:17:14: Event postcondition 'balances[to] ==
before(balances[to]) + amount' might not hold before emit
transferred.
Errors were found by the verifier.
```

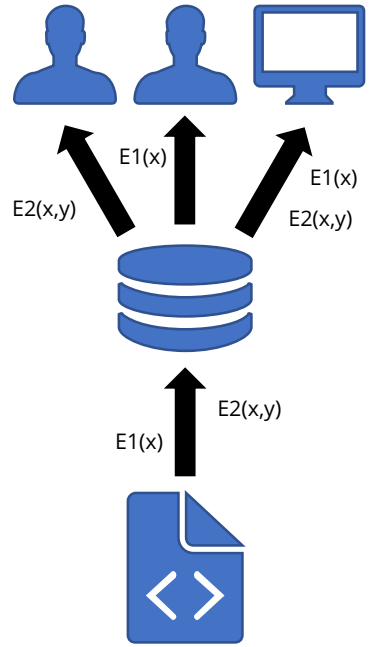
Example

```
contract Token {
  mapping(address=>uint) balances;

  /// @notice tracks-changes-in balances
  /// @notice precondition  balances[from] >= amount
  /// @notice postcondition balances[from] == before(balances[from]) - amount
  /// @notice postcondition  balances[to] == before(balances[to]) + amount
  event transferred(address from, address to, uint amount);

  /// @notice emits transferred
  function transfer(address to, uint amount) public {
    require(balances[msg.sender] >= amount && msg.sender != to);
    balances[msg.sender] -= amount;
    balances[to] += amount;
    emit transferred(msg.sender, to, amount);
  }
}
```

```
$ solc-verify.py SimpleToken.sol
Token::transfer: OK
No errors found.
```

Conclusions

- Solidity events provide abstract view
- Formal specification and verification
  - In-code annotations
  - Instrumentation



youtu.be/NNytwVBZ1no

arxiv.org/abs/2005.10382

github.com/SRI-CSL/solidity

hajduakos.github.io

```
contract Token {
  mapping(address=>uint) balances;

  /// @notice tracks-changes-in balances
  /// @notice precondition  balances[from] >= amount
  /// @notice postcondition balances[from] == before(balances[from]) - amount
  /// @notice postcondition balances[to] == before(balances[to]) + amount
  event transferred(address from, address to, uint amount);

  /// @notice emits transferred
  function transfer(address to, uint amount) public {
    require(balances[msg.sender] >= amount && msg.sender != to);
    balances[msg.sender] -= amount;
    balances[to] += amount;
    emit transferred(msg.sender, to, amount);
  }
}
```