Isabelle/Solidity for Smart Contracts

Jane Open Access
Dummy University Computing Laboratory, [optional: Address], Country
My second affiliation, Country

Joan R. Public[1]
Department of Informatics, Dummy College, [optional: Address], Country

Abstract

2012 ACM Subject Classification Replace ccsdesc macro with valid one

Keywords and phrases Program Verification, Smart Contracts, Isabelle, Solidity

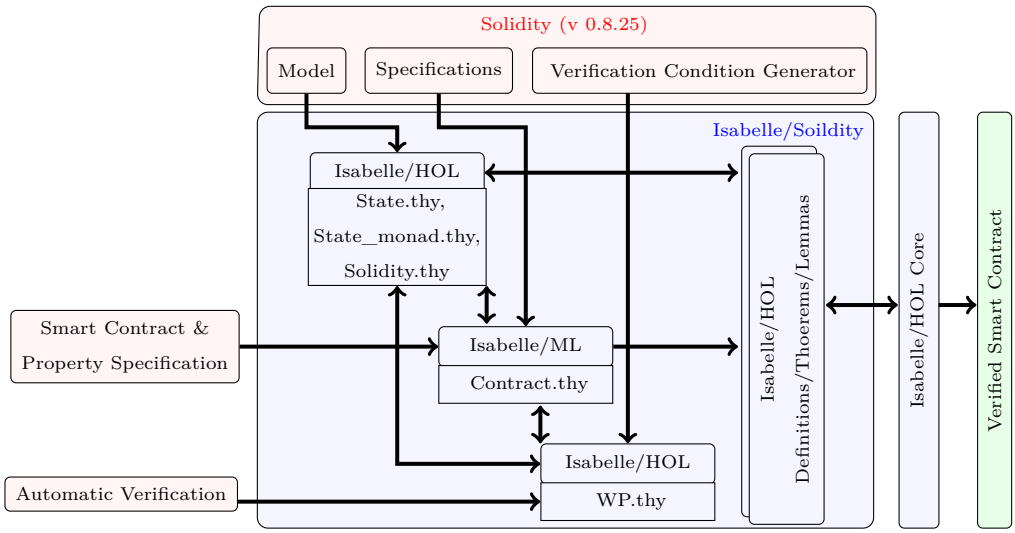
Digital Object Identifier 10.4230/OASIcs.CVIT.2016.23

Funding *Jane Open Access*: (Optional) author-specific funding acknowledgements
Joan R. Public: [funding]

Acknowledgements I want to thank ...

1 Introduction

2 Overview



[1] Optional footnote, e.g. to mark corresponding author

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:3
OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

3 Case Study

Listing 1: Solidity source code for the Casino

```
1  contract Casino {
2    enum Coin { HEADS, TAILS } ;
3    enum State { IDLE, GAME_AVAILABLE, BET_PLACED }
4    State private state;
5    address public operator, player;
6    uint public pot;
7    bytes32 public hashedNumber;
8    uint public bet;
9    Coin guess;
10
11   function createGame(bytes32 hashNum)
12   public byOperator, inState(IDLE) {
13   hashedNumber = hashNum;
14   state = GAME_AVAILABLE;
15   }
16
17   function placeBet(Coin _guess) public payable inState(GAME_AVAILABLE) {
18   require (msg.sender != operator);
19   require (msg.value <= pot);
20   state = BET_PLACED;
21   player = msg.sender;
22   bet = msg.value;
23   guess = _guess;
24 }
25
26   function decideBet(uint secretNumber)
27   public byOperator, inState(BET_PLACED) {
28     require (hashedNumber == keccak256(secretNumber));
29     Coin secret = (secretNumber % 2 == 0)? HEADS : TAILS;
30     if (secret == guess) { pot = pot - bet; player.transfer(bet*2); bet =
           0;}
31     else {
32     pot = pot + bet; bet = 0;
33       }
34   state = IDLE;}
35   function addToPot() public payable byOperator { pot = pot + msg.value;}
36
37   function removeFromPot(uint amount) public byOperator, noActiveBet {
         operator.transfer(amount); pot = pot - amount;}
38   }
```

Listing 1 is a Solidity source code for Casino smart contract from verifyThis competition. The contract has three explicit states: `IDLE`, `GAME_AVAILABLE`, `BET_PLACED` (Line 3). An operator may creat a new game by calling `creatGame` function in `IDLE` state. The `creatGame` function uses two modifiers (`byOperator` and `inState(s)`) to ensure that only operator can creat the game in `IDLE` state. Moreover, operator also needs to provide `hashNum` value which is later used as a unique reference in deciding the outcome of the created game (Line 28) in `decideBet` function. Finally, in `IDLE` state, an operator may add or remove any amount of money by invoking `addToPot` or `removeFromPot` function.

Once the game has `GAME_AVAILABLE` state, a player may place a bet by calling `placeBet` function. A player can pass his bet on `HEAD` or `TAIL`, as a `_guess`, in `placeBet` function. The

function provides safegurad using `require` from operator placing the bet and amount of bet exceeding pot balance. The function changes the state to `BET_PLACED`. In `GAME_AVAILABLE` state, too, an operator may add or remove any amount of money by invoking `addToPot` or `removeFromPot` function..

In `BET_PLACED` state, the operator may decide the bet using `decideBet` function. The function uses `secreteNumber` to verify the `hashNumber` generated when game was created. The function, then compares the player's bet and transfers the double the amount of bet in case of right guess otherwise draws the amount from the pot to operator account.

4 Specification

5 Related Work

6 Conclusion