Formal Verification of Solidity Smart Contracts Using Dafny

Alireza Lotfi Takami
Krishna Kantwala
Aditi Patade

University of Waterloo

August 24, 2020

Overview

Introduction

- Smart Contract: A program that is written and uploaded on to the Ethereum blockchain
- Distributed Application: One or more smart contracts that interact with each other to create an application on the Ethereum blockchain
- The code controls the execution, and transactions are trackable and irreversible.
- Once a smart contract is put into a block, it is given an address and has the ability to act like a person
- The smart contract cannot be changed once running on the blockchain.

Purpose

- Smart contracts-
  - Written in a Turing-complete programming language - Solidity
  - What guarantees can we make about its logic/soundness?
- Problem:
  - Smart Contract usually deal with money, and once uploaded to the blockchain can never be altered.Therefore, hard to fix in the case of bugs.
- Static analysis is very important to this field.

Our Method

To ensure bug-free smart contract, we propose-

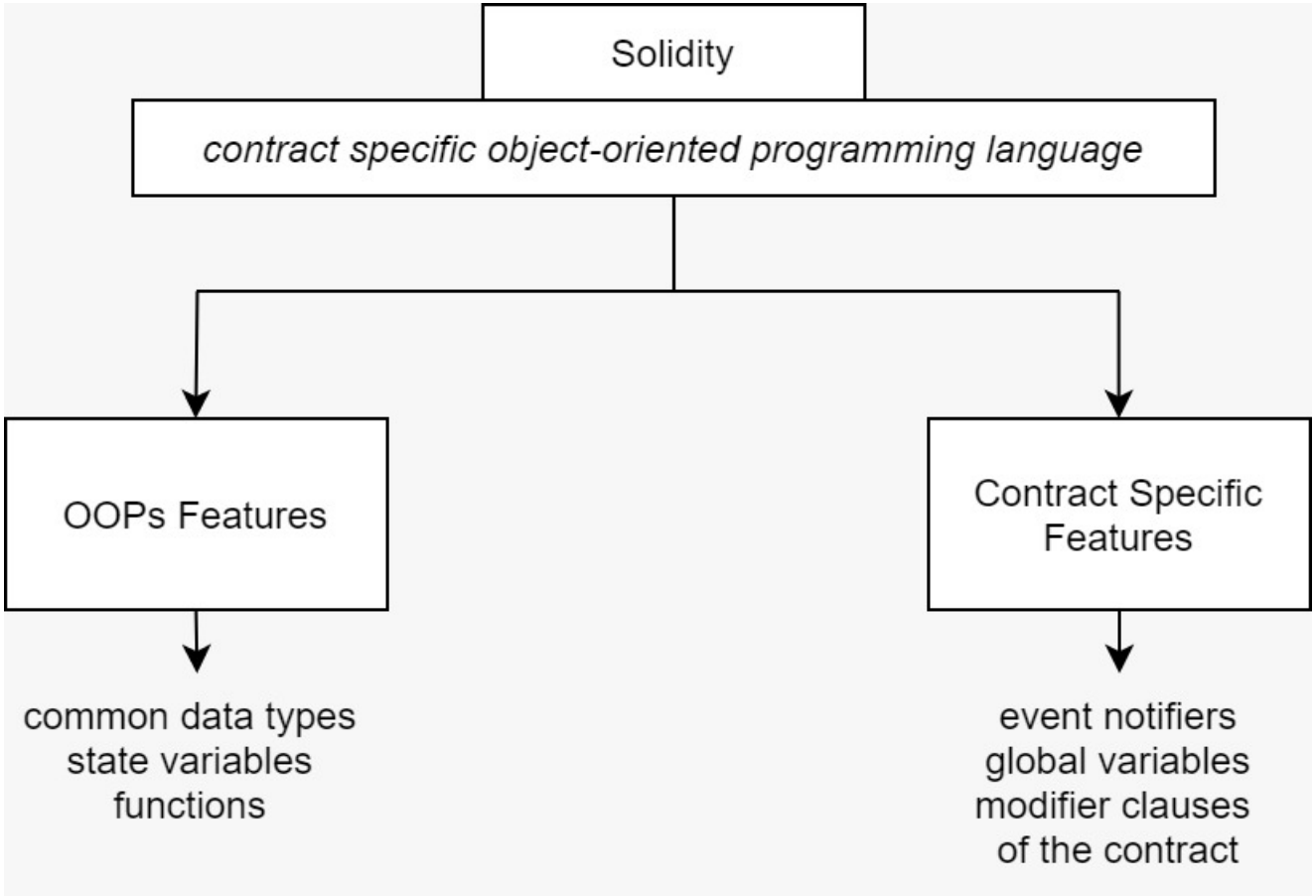
- Command line tool
  - verify the smart contract using static verification
  - contracts will be parsed using ANTLR which will generate the parse tree which will be further translated to to another programming language called Dafny as it support formal verification, using Python.

Toolchain

Solidity

1. A programming language which is used for implementing smart contracts
2. Using Solidity, one can write applications such as voting, crowdfunding, blind auctions, and multi-signature wallets etc.
3. They are compiled to bytecode in the Ethereum blockchain for the execution without requiring centralized or trusted parties.

Solidity


Solidity
contract specific object-oriented programming language
OOPs Features
Contract Specific Features
common data types
state variables
functions
event notifiers
global variables
modifier clauses
of the contract

ANTLR - Another Tool for Language Recognition

- A tool which helps to create parser for-
  - reading
  - processing
  - executing
  - translating
- Using ANTLR, one can generate **a parser that can build and walk parse trees in their specific target language**.
- It helps to **define a grammar for the language, data format, diagram or any kind of structure** that is represented with text for analyzing.

Dafny

- An imperative object-oriented programming language
- Used to verify the functional correctness of programs with the help of built-in automatic program verifier.
- Dafny is using built-in specification such as -
  - preconditions
  - post-condition
  - frame specifications
  - termination metrics
  - loop variants and invariant

  to support formal verification of program, which helps to prove correctness of code.

Results

- Antlr was feed with solidity code and a proper grammar file.
- Antlr uses an input code and its proper grammar and creates a parse tree which can be used for any purpose.
- As Python decreases the development time, python was used as Antlr target language.


```
{'children': [{'name': 'solidity',
               'type': 'PragmaDirective',
               'value': '>=0.4.22<0.6.0'},
              {'baseContracts': [],
               'kind': 'contract',
               'name': 'test_contract',
               'subNodes': [{'body': {'statements': [{'number': '3',
                                                      'subdenomination': None,
                                                      'type': 'NumberLiteral'}],
                                      'type': 'Block'},
                             'isConstructor': False,
                             'modifiers': [],
                             'name': 'ifalone',
                             'parameters': {'parameters': [{'isIndexed': False,
                                                            'isStateVar': False,
                                                            'name': 'y',
                                                            'storageLocation': None,
                                                            'type': 'Parameter',
                                                            'typeName': {'name': 'int',
                                                                         'type': 'ElementaryTypeName'}}],
                                            'type': 'ParameterList'},
                             'returnParameters': {'parameters': [{'isIndexed': False,
                                                                  'isStateVar': False,
                                                                  'name': None,
                                                                  'storageLocation': None,
                                                                  'type': 'Parameter',
                                                                  'typeName': {'name': 'int',
                                                                               'type': 'ElementaryTypeName'}}],
                                                  'type': 'ParameterList'},
                             'stateMutability': None,
                             'type': 'FunctionDefinition',
                             'visibility': 'public'}],
               'type': 'ContractDefinition'}],
 'type': 'SourceUnit'}
```

Results (contd)

- A command line user interface was created in python which used ANTLR's output parse tree to translate them to Dafny for verification of solidity smart contracts.

```
pragma solidity >=0.4.0 <0.7.0;

contract test_contract {

    function ifelse (int x) public returns (int) {
        if(x == 0) {   // if else statement
            result = 1;
        } else {
            result = 2;
        }
        return result;
    }
}
```

(a) Solidity Code

```
method ifelse(x: int) returns (result : int)
requires x >= 0
{
  if(x == 0){
    result :=1 ;
  }
  else{
    result :=2 ;
  }
  assert result > 0;
  return result;
}
```

(b) Dafny Code

Figure: Verification of a solidity smart contract function with one if-else.

Results (contd)

```
pragma solidity >=0.4.0 <0.7.0;

contract test_contract {
    function ifelsenested (int x) public returns (int) {
        if(x < 10) {
            if (x == 8){
                result = 9;
            }
            else {
                result = 10;
            }
        } else {
            result = 11;
        }
        return result;
    }
}
```

(a) Solidity Code

```
method ifelsenested(x : int) returns (result : int)
ensures result > 7
requires x > 6
{
   if(x < 10) {   // if else statement
     if(x == 8) {   // if else statement
        result := 9;
      } else {
                result := 10;
            }
     }
   else {
            result := 11;
        }
   return result;
}
```

(b) Dafny Code

Figure: Verification of a solidity smart contract function with nested if-else.

Future Work

- Handle all solidity syntax with the proposed tool.
- Feature - Using Nusmv to do the verification for global and state variables.