UNIVERSITY OF WATERLOO
ECE653 - SOFTWART TEST/QUAL ASSUR/MAINT
PROJECT PROPOSAL

# Formal Verification of Solidity Smart Contracts Using Dafny

***Report by:***
Alireza LOTFI TAKAMI
*Student ID:* 20854299
*Email:* alotfita@uwaterloo.ca
Krishna D KANTWALA
*Student ID:* 20868348
*Email:* kdkantwa@uwaterloo.ca
Aditi S PATADE
*Student ID:* 20868672
*Email:* apatade@uwaterloo.ca

***Instructor:***
Dr. Vijay GANESH

May 29, 2020

# 1 Abstract

A smart contract is a self-executing contract with the terms of the agreement between buyer and seller being directly written into lines of code. The code and the agreements contained therein exist across a distributed, decentralized blockchain network. The code controls the execution, and transactions are trackable and irreversible. Smart contracts permit trusted transactions and agreements to be carried out among disparate, anonymous parties without the need for a central authority, legal system, or external enforcement mechanism [1].

The goal of this project is to develop a tool to ensure bug-free designing of the smart contract and verifying its reachability property. This tool is expected to take the smart contract wriiten in solidity and translate it to dafny. Dafny being a static program verifier will be used to verify the functional correctness of the contract [2]. We consider correctness as the state of the program without any bugs such that the program observes safe programming practices.

# 2 Problem Statement

Smart contracts are executable programs that enable the building of a programmable trust mechanism between multiple entities without the need of a trusted third-party. Smart contracts are often written in a Turing-complete programming language called Solidity, which is not easy to audit for subtle errors [see Abstract][3]. Although the faithful execution of a smart contract is enforced by the consensus protocol of the blockchain, it remains the prerogative of the participating organisations to check the validity of the smart contract , i.e., the syntactic implementation meets the best practises, and validate its fairness, i.e. the code adheres to the agreed higher-level business interaction logic [4]. Therefore, to ensure bug-free smart contract we are developing a framework which will primarily focus on guaranteeing the bug free smart contract.

# 3 Approach

Our approach consist of using smart contracts written in solidity as a best case for the tool. These contracts will be used as input for the parser tool- ANTLR which will help us translate the input code to dafny. Dafny being a supporter of formal verification tool will verify the translated code, thus giving an output in the form of a report with a list of detected error, if any.

# 4 Tool Chain

We are going to use different tools in our project. In this section we are going to briefly explain each of the tool.

### 4.1 Smart contracts

Smart contracts are the building blocks that are used to create decentralized application.

### 4.2 Solidity

Solidity is a programming language which is used for implementing smart contracts. With the help of Solidity, we can write applications that implement self-enforcing business logic such as voting, crowdfunding, blind auctions, and multi-signature wallets. Even we can say that solidity is smart contract specific language [5].

Smart contracts which are implemented in solidity language are self-executing piece of code that are compiled and pushed to the Ethereum blockchain [6]. Solidity program is complied to bytecode that is executable on the EVM (Ethereum virtual machine). EVM is basically a runtime environment for smart contracts in Ethereum.

### 4.3 Antlr

Antlr (Another Tool for Language Recognition) is a powerful parser generator, a tool which helps to create parser for reading, processing, executing or translating structured text or binary files. To parse a typical program using regular expression is not enough specially when program is using recursion but with the help of Antlr it becomes easier, faster and mess free. Antlr is used to build languages, tools, and frameworks. With the help of Antlr, one can generate a parser that can build and walk parse trees in their specific target language. Antlr helps to define a grammar for the language, data format, diagram or any kind of structure that is represented with text for analyzing [7].

### 4.4 Dafny

Dafny is an imperative object-oriented programming language which is used to verify the functional correctness of programs with the help of built-in automatic program verifier. Dafny is using built-in specification construct to support formal verification of program. These specifications include preconditions, postcondition, frame specifications, termination metrics, loop variants and loop invariants which helps to prove correctness of code [2]. With the help of Dafny, it becomes easier for programmers to write correct code without having any runtime errors and also correct in terms of what exactly program intended to do.

## 5 Benchmark

As we explained before, in this project our goal is to define some queries in smart contract solidity code and turn the smart contract solidity code and it's defined queries into Dafny code. In order to test our project we are going to create some basic smart contracts

and define different queries inside them. These smart contracts are our project test cases. We are going to use different solidity codes to cover most of the solidity syntax.

## 6 Plan of Action and Feasibility Analysis

In order to have meaningful individual contribution from all the team member, the members need to get familiar with the Tool chains- ANTLR, Solidity, Python/Java, and Dafny. As per our estimation, this step is expected to be completed within two to three weeks time frame. Once the tool- familiarization step is completed, we shall begin with a simple smart contract using solidity. Simultaneously, we shall be writing a Python or Java code which uses ANTLR to parse the solidity smart contract. Using ANTLR, we are planning to design a parser which is expected to recognize different formats of the smart contract and create necessary rules which are required for the parser. The output of this parser code shall be a Dafny code, if no bugs are encountered. Since we shall be starting with a simple smart contract as an input of our project, we anticipate to do this in two weeks and get the first output of our project in week five of the project. Therefore, we have about five weeks to add more features to our project. We are planning to support a new syntax of solidity in each step of the project. As per our timeline estimation, by the end of ten weeks we are expecting an executable code which shall formally verify the smart contract, thus covering most of the solidity syntax. Our benchmark also consist of working on considerable number of smart contracts.

## 7 Demo

Our final application will be a command line application which takes solidity program file as an input and gives reachability and unreachability of queries inside solidity as an output.

## References


[1] "Smart contracts definition," https://www.investopedia.com/terms/s/smart-contracts.asp, (Accessed on 05/27/2020).

[2] "Dafny: A language and program verifier for functional correctness - microsoft research," https://www.microsoft.com/en-us/research/project/dafny-a-language-and-program-verifier-for-functional-correctness/, (Accessed on 05/27/2020).

[3] Zhang, William and Banescu, Sebastian and Pasos, Leodardo and Stewart, Steven and Ganesh, Vijay, "MPro: Combining Static and Symbolic Analysis for Scalable Testing of Smart Contract," *arXiv preprint arXiv:1911.00570*, 2019.

[4] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, "Zeus: Analyzing safety of smart contracts." in *NDSS*, 2018, pp. 1–12.

[5] "Solidity — solidity 0.6.8 documentation," https://solidity.readthedocs.io/en/v0.6.8/, (Accessed on 05/26/2020).

[6] "Solidity - wikipedia," https://en.wikipedia.org/wiki/Solidity, (Accessed on 05/26/2020).

[7] "Antlr," https://www.antlr.org/, (Accessed on 05/26/2020).