Smart Contract Platform Design: Verification through Formal Methods

I. Introduction

With the rising popularity of and buzz surrounding bitcoin, the underlying technologies supporting the e-currency have been revitalized as well. The transparent and distributed ledger for maintaining transactional data (known as blockchain) as well as the decentralized consensus protocol used for updating such ledgers carry over to the recent development of smart contracts. This concept originated with work from Nick Szabo in 1996 ("Smart Contracts: Building Blocks for Digital Markets") and is most notably implemented by Ethereum, a platform that provides a virtual machine (the Ethereum Virtual Machine, or EVM) which may execute scripts using a network of nodes distributed internationally.

However, while the excitement surrounding blockchain and smart contracts continues to rise (reaching the realms of information technology, finance, and even government), significant vulnerabilities in the design of such software have become apparent. Some lead to significant losses, such as in “The DAO Attack,” a case in which a Ethereum-supported crowdfunding service lost roughly 50 million USD of investments due to a bug in the program which was exploited by hackers.

Thus, as these technologies work their way into global finance, transportation, healthcare, and numerous other industries and sectors, it is imperative that smart contract platforms are designed to combat existing vulnerabilities as well as broader categories of bugs. This project seeks to verify and analyze implementations of smart contracts, namely in the traditions of the Ethereum platform.

II. Proposed Structure

In this project, I will be using formal verification methods to reason about the security of smart contract platforms. The first component of this project involves research into the conventions of smart contract technology and understanding which components
may be especially prone to vulnerabilities (for example, from preliminary research, the
theory behind blockchain requires truly stochastic processes, which practically is difficult
to implement on-chain). Other commonalities in contract design which may be explored
in this project include implementations of contract authorization, termination, forking,
and oracles.

The second component of this project is the production of a proof which reasons
about whether (or not) elements of a smart contract platform are secure. For this I will be
using Coq, a proof management language well familiar to the Yale FLINT Group under
Professor Shao, and used in research on smart contracts by Vilhelm Sjöberg and another
undergraduate researcher, Christopher Fu. These contracts are written in DeepSEA,
which generates Coq code that I will produce formal proofs for with direction from
Vilhelm Sjöberg. The project will strive to form conclusions concerning higher-level
functionalities and features of the smart contract system, rather than lower-level systems
software components like the compiler. The aim is to verify code that can be run on the
Ethereum Virtual Machine.

III. Deliverables
At the end of term, I will provide the following:

○ A report (written) that details the research and organization from the first part of
  the project, i.e. notable patterns in smart contract design and my interpretations of
  their relative accessibility as targets of formal verification.
○ A report (presentation-style) that details the process undertaken to formulate the
  proof, including exploration of formal verification theory and testing.
  Furthermore, this report will explain the significance of any conclusions that can
  be derived from the content of the project.
○ Relevant proof scripts, Coq-compatible

IV. Timeline

Note: I will be organizing my work in two-week sprints, detailed below.

○ Week of Jan. 22, Jan. 29
  ■ Re-familiarize with software analysis and computer-assisted theorem
    proving, the Coq proof assistant, functional programming, op. semantics,
    higher-order logic
  ■ Study existing literature on blockchain/smart contracts, Ethereum
○ Week of Feb. 5, Feb. 12
  ■ Familiarization with Solidity (a programming language for implementing smart contracts, primarily developed by Ethereum’s team)
  ■ Explore documented vulnerabilities of smart contracts
  ■ Identify potential sites of analysis
○ Week of Feb. 19, Feb. 26
  ■ Begin organization/write-up of notes from previous weeks
  ■ Work on proof begins
○ Week of Mar. 5, Mar. 12
  ■ Proofs cont., build on write-up
○ Week of Mar. 19, Mar. 26
  ■ Proofs cont., build on write-up
○ Week of Apr. 2, Apr. 9
  ■ Write-up and begin work on presentation
○ Week of Apr. 16, Apr. 23
  ■ Complete reports from write-up content, produce presentation slides and verbal content
○ Week of Apr. 30, May 6
  ■ Miscellany; polishing completed project

V. Calibration

○ I intend to meet with Professor Shao regularly (on a per-month basis) to update him on my progress and re-calibrate the timeline and goals of this project as necessary.
○ I intend to meet and correspond with Vilhelm Sjöberg regularly to discuss the technical progress of the research.