Abstract

Ethereum is a public, open-source platform that allows users to create contracts that can store data on the Ethereum blockchain and send and receive messages to and from other contracts. Such contracts can be combined in complex ways to create powerful decentralized applications. Because of the nature of the Ethereum blockchain, bugs in contract code are difficult to rectify and may lead to large monetary losses if exploited. For this project, we propose to extend the compiler for DeepSEA, a certified programming language used in the development of mCertiKOS, to generate Coq specifications from DeepSEA specifications of Ethereum contracts written in Solidity.

1 Introduction to Ethereum

Ethereum is a public, open-source platform that allows users to build and run decentralized applications that run on blockchain technology [3]. A blockchain is a distributed computing architecture where each node in the network executes and records transactions as blocks in the blockchain. Because only one block is added at a time and each block contains a cryptographic proof of its validity, each node in the network can agree upon the current state of the blockchain.

The blockchain forms the backbone of Bitcoin, a cryptocurrency released in 2009, which uses it to record transactions between users [4]. Ethereum, on the other hand, keeps track of the state of every account on the blockchain.

The account is the basic building block of Ethereum, of which there are two types: externally owned accounts (EOAs) and contract accounts (contracts) [1][2]. EOAs are human-controlled because they are
controlled by private keys held by the accounts owner. Contracts (often also referred to as \textit{smart contracts}), on the other hand, are a collection of code and data that reside at a specific address on the Ethereum blockchain in a binary format called Ethereum Virtual Machine (EVM) bytecode. Contracts, which are typically written in a higher-level language like Solidity and compiled into EVM bytecode to be deployed to the blockchain, can send and receive messages to and from each other and to accounts. In this manner, they can be put together to build decentralized applications backed by the Ethereum blockchain. Examples of such applications range from voting applications, to peer-to-peer trading markets, to video games.

2 Motivation

Contracts are such a young and developing technology that their interactions can lead to unintentional consequences and losses of large amounts of money. One such event was the Decentralized Autonomous Organization (DAO) hack, which occurred in June 2016 \cite{5}. The DAO was a set of Ethereum contracts that enabled people to buy tokens representing voting rights during a funding period, called an initial coin offering (ICO) \cite{6}. Once the ICO was over, investors could vote on proposals on how the DAO should spend the money. By May 2016, DAO had raised $160 million in Ether, the token used by Ethereum clients to pay for computations on the EVM, from thousands of anonymous investors. But just one month later, a hacker was able to exploit a bug in the DAO contract code to siphon away 3.6 million Ether ($50 million)—more than a third of the 11.5 million Ether in the DAO at the time.

3 Proposal

We propose to use DeepSEA, a certified programming language developed to allow developers to write specifications for program behavior \cite{7}, to write formally verified smart contracts. This way, we can guarantee that the behavior of such contracts will be well-defined and bug-free. DeepSEA was originally used to aid in the development of mCertiKOS, a fully verified hypervisor that can boot a version of Linux as a guest. Edsger, the DeepSEA compiler frontend, and a Coq-CompCertX backend are used to compile DeepSEA specifications into certified executables. Coq is a formal proof assistant that provides a language to write programs and prove properties about them, and CompCertX is a variant of CompCert, a formally verified C compiler that can generate code for x86, PowerPC, and ARM processors.
For this project, we wish to generate Coq specifications from DeepSEA specifications of Solidity contracts. This will allow us to prove properties of DeepSEA contract specifications in Coq. We will need to extend the existing DeepSEA frontend to generate an intermediate representation (IR) that sufficiently models a small but useful subset of Solidity. In order to devise such an IR, we will first look at the structure of typical Solidity contracts, such as how messages are handled and how contracts call each other. Once we have modeled a few such example contracts in Coq, we can then determine some useful properties to prove about contracts. One simple example for a contract that allows users to exchange fixed-supply tokens is that the total number of tokens in users’ possession is indeed constant for all sequences of transactions. More interesting properties will become clear once we model multi-contract applications. The hope is that, once the DeepSEA frontend has been modified to automatically generate Coq specifications, we obtain a useful framework for proving properties about contracts in Coq.

The ultimate goal is to be able to compile DeepSEA specifications into an EVM executable as well. This would require porting the Coq-CompCertX backend to generate EVM bytecode. This is outside the scope of our project, and we will only be focusing on extending the frontend of the DeepSEA compiler.

4 Deliverables

(a) Model a few contracts written in a simple subset of Solidity in Coq.

(b) Determine and prove interesting properties about these contracts.

(c) Devise an intermediate representation for DeepSEA specifications that can describe that subset of Solidity.

(d) Extend the DeepSEA compiler to generate Coq specifications of Solidity contracts.

References


