Static Resource Analysis of Smart Contracts – Milestone 2

Nick Roberts
Carnegie Mellon University
nroberts@andrew.cmu.edu
January 31, 2018

1 Summary of updates

Since last update, I have decided which intermediate form for OCaml is the most suitable for compilation into EVM bytecode. This intermediate representation is similar to the execution model of EVM, and so I expect to be able to compile a larger subset of OCaml than originally planned. However, I have delayed the milestone of providing an OCaml interface to EVM primitives, since the core OCaml compiler is more central to the goals of this project.

2 Accomplishments

Continuing work from last semester, I have extensively reviewed the Ethereum yellow paper [1] and updated the small compiler for arithmetic expressions I wrote last semester. I successfully ran contact creation code emitted by my compiler on `javaethereum`, a Java implementation of the EVM.

I have identified the `instruct` intermediate form of the OCaml compiler as the one which I should compile to the EVM. This form corresponds closely to that outlined in Xavier Leroy's paper [2]. I have read this paper and watched related talks to understand the differences between this execution model and that defined by EVM.

2.1 Meeting the milestone

My milestone as stated in my proposal was:

> Write module for direct access to EVM primitives from OCaml. Compile all arithmetic expressions from OCaml to EVM, not including user-defined data types.

I have written the compiler for arithmetic expressions, so this part of the goal is clearly met. Instead of the module for direct access to EVM primitives, I have focused instead on the mapping between the ZINC abstract machine and the EVM, which is the avenue I will continue to pursue for the upcoming milestone.

2.2 Surprises

Solidity, an existing language that compiles to the EVM, has defined its own ad hoc protocol for procedure calls. Its protocol describes procedure calls both within the currently-executing project and in other contracts. Since Solidity is so popular, and since interoperability with Solidity is a desirable goal for us, we intend to adopt the same protocol in our compiler.

This protocol, although surprising, does not appear to derail our design for the compiler. Arguments are passed on the stack, like ZINC, and the inter-contract protocol is not complicated. More than anything, this protocol serves as a reminder that our design decisions are constrained by design decisions of past implementations.

3 Looking ahead

3.1 Revisions to future milestones

For the upcoming milestone, I am still focusing on procedure calls, but in the context of translating the ZINC abstract machine to the EVM. I am writing an interpreter for the `instruct` intermediate form of the OCaml compiler so that I can understand the relationship between it and ZINC.

3.2 Resources needed

I already have the OCaml compiler source code installed, so I am good for now.

References

[1] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger eip-150 revision (759dccd - 2017-08-07), 2017. Accessed: 2018-01-03.

[2] Xavier Leroy. The ZINC experiment: an economical implementation of the ML language. Technical report 117, INRIA, 1990.