Static Resource Analysis of Smart Contracts – Milestone 3

Nick Roberts
Carnegie Mellon University
nroberts@andrew.cmu.edu
February 14, 2018

1 Summary of updates

For this milestone, I continued the work I began at last milestone: writing an interpreter for a phase in the OCaml compiler. This phase has a close-but-not-exact correspondence with the execution model of the EVM, and so I will work to bridge this gap before the next milestone meeting.

2 Accomplishments

Since last milestone, I have written an interpreter in OCaml for a significant portion of the bytecode emitted by the `ocamlc` compiler. The subset I have covered includes arithmetic, variable storage, function application, and closure creation. This interpreter uses representations that correspond closely to the model of the EVM, since both are stack-based with access to (for all practical purposes) unlimited heap memory.

You can track my progress with the interpreter on Github.[1]

2.1 Meeting the milestone

In my previous milestone, I identified that for this milestone, I would write an interpreter for OCaml bytecode up to function application. I have accomplished this, and I additionally have included closure creation. I have decided that my bytecode interpreter will work with version 4.06 of `ocamlc`.

2.2 Surprises

The documentation for OCaml's bytecode is sparse and inconsistent. I found a summary of the instruction set as of version 3.11.2 of `ocamlc` online, but this document[2] does not fully describe the effect of an instruction on the execution state. Luckily, the C implementation of OCaml's bytecode interpreter is reasonably clearly written.[3]

3 Looking ahead

3.1 Revisions to future milestones

For next milestone, I will write a new backend to the OCaml compiler that outputs EVM bytecode instead of OCaml bytecode. To do this, I will start from the OCaml interpreter I have already written and determine how to map my representation onto the primitives of the EVM. There are some representational differences between the dynamics of OCaml bytecode and EVM bytecode: for instance, the OCaml bytecode interpreter includes an accumulator value

[1] `https://www.github.com/ncik-roberts/ocaml-interpreter`
[2] `http://cadmium.x9c.fr/distrib/caml-instructions.pdf`
[3] `https://github.com/ocaml/ocaml/blob/trunk/byterun/interp.c`

separate from the stack, and this concept is not primitive to the EVM. For these, I will have to carefully determine a sound translation.

This milestone goal diverges from what I originally proposed, which was to incrementally add OCaml features to my EVM bytecode compiler. Instead, I am converting a large portion of OCaml bytecode to EVM bytecode, which supports many OCaml features for free, including algebraic datatypes and closures. This is exciting, because it wasn't originally clear if such features were in the scope of this project.

3.2 Resources needed

None beyond what I already have.