

# Specifying the Ethereum Virtual Machine for Theorem Provers

Yoichi Hirai

Ethereum Foundation

Cambridge, Sep. 13, 2017  
(FC 2017 + some updates)

# Outline

## Problem

- Motivation

- EVM as a Machine

- Wanted Properties

## Current Efforts

- Writing Specification

- Testing the Specification

- Proving Ethereum Contracts Correct

## Outlook

# Outline

## Problem

### Motivation

EVM as a Machine

Wanted Properties

## Current Efforts

Writing Specification

Testing the Specification

Proving Ethereum Contracts Correct

## Outlook

# Ethereum: Program Execution without Trusted Admin

“Server side” computation dictates the society now.  
Computers have owners and administrators.

- ▶ Will my program be executed unmodified?
- ▶ Will my program be available?
- ▶ Will my data kept secure from unauthenticated modification?

Ethereum currently uses a Bitcoin-like approach

1. to replicate programs and program states, and
2. to agree on execution traces.

Over 24,000 nodes<sup>1</sup> are running a clone of the Ethereum Virtual Machine (EVM).

---

<sup>1</sup>According to [ethernodes.org](http://ethernodes.org).

# Typical Ethereum Usage: Deposits & Announcements

**Ethereum Name Service** is a sealed second-price auction.

The price is **locked** while the name is held.

Roughly 168,000 ETH ( $\approx$  42,000,000 GBP) locked for 161,000 names.

**Voting Protocol** McCorry, Shahandashti and Hao [FC 2017] implemented a voting protocol on Ethereum.

The protocol requires a **public bulletin board**; and uses **deposit** to incentivize participants to perform all steps.

Counterparty risks are now on programs (“smart contracts”).  
At least you can read the code. Isn't that enough?

# The Famous Bug

“The DAO” (an investment club): funds moved out unexpectedly.

17% of total existing ETH affected.

Many miners<sup>2</sup> accepted a protocol change to remedy this particular case; the network split.

The EVM didn't have a problem; the program on top had.

## EVM might be a Good Formalization Target, I Thought

- ▶ unstoppable app sounds crazy unless it's proven correct
- ▶ easy machine (deterministic on all inputs)
- ▶ test cases for multiple implementations
- ▶ a short spec (33 pages).

---

<sup>2</sup>Miners run GPUs to produce valid blocks.

# EVM turns out not too Big to Formalize

The EVM definition in Lem (an ML like specification language) has 2,000 lines.

Most instructions are simply encoded as functions in Lem:

Yellow Paper (original spec):

0x06 MOD

2 1

Modulo remainder operation.

$$\mu'_s[0] \equiv \begin{cases} 0 & \text{if } \mu_s[1] = 0 \\ \mu_s[0] \bmod \mu_s[1] & \text{otherwise} \end{cases}$$

Lem:

```
| Arith MOD -> stack_2_1_op v c
  (fun a divisor -> (if divisor = 0 then 0 else
    word256FromInteger ((uint a) mod (uint divisor))
  ))
```

# Outline

## Problem

Motivation

**EVM as a Machine**

Wanted Properties

## Current Efforts

Writing Specification

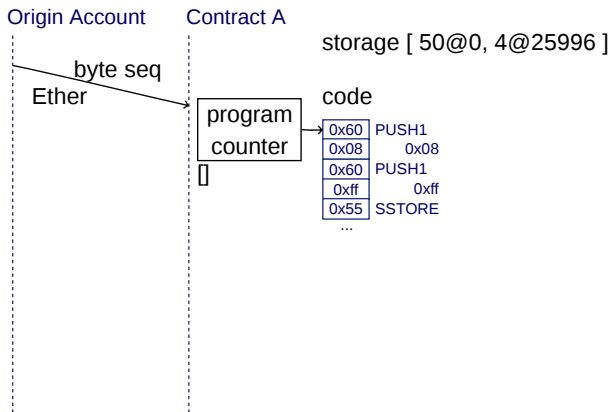
Testing the Specification

Proving Ethereum Contracts Correct

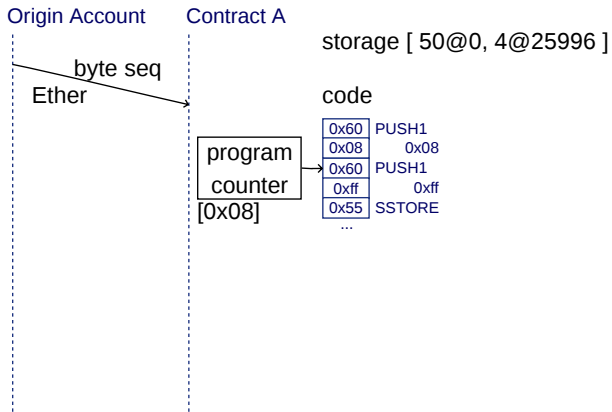
## Outlook



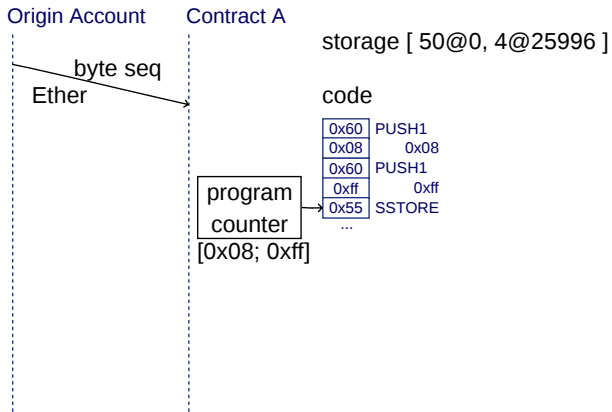
# How EVM Works 1



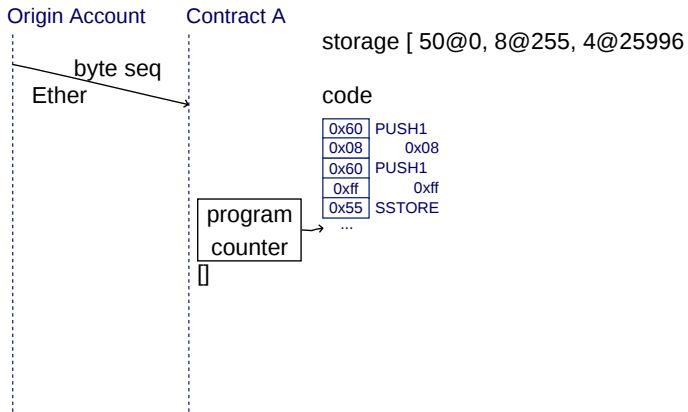
# How EVM Works 2



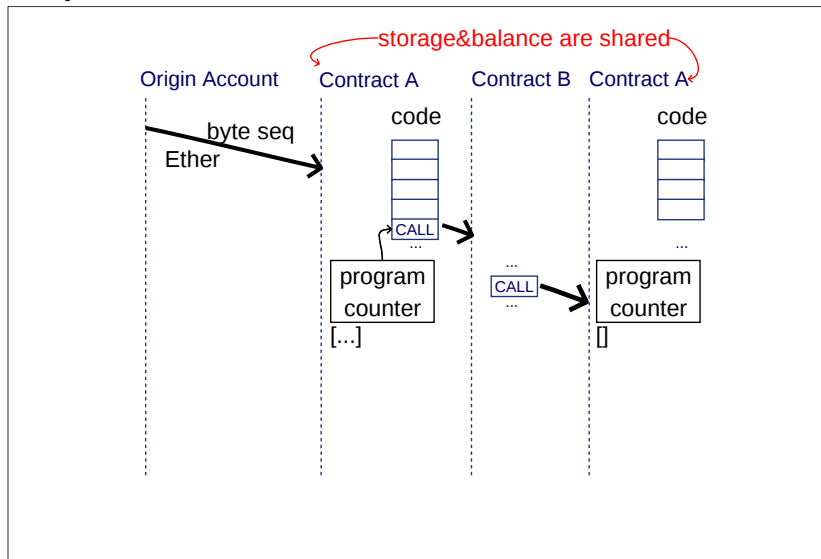
# How EVM Works 3



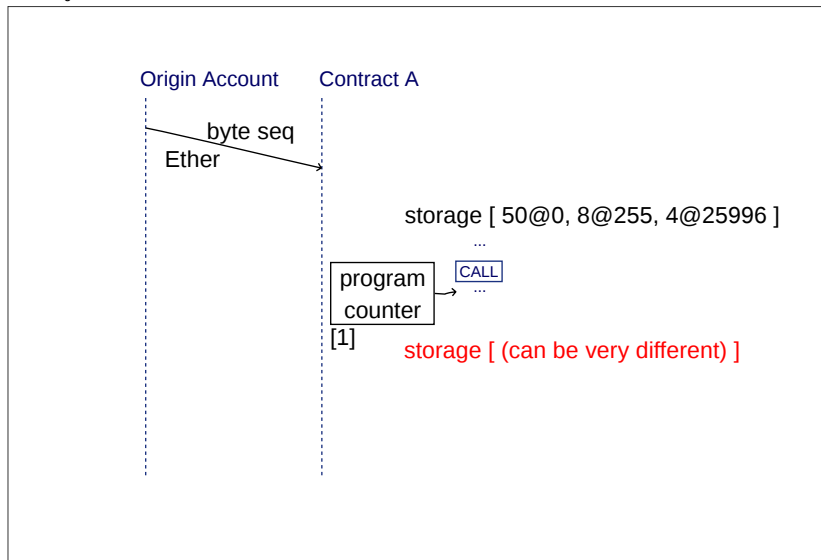
# How EVM Works 4



# An Annoying Phenomenon Called Reentrancy (Transaction's View)



# An Annoying Phenomenon Called Reentrancy (Invocation's View)



# Outline

## Problem

Motivation

EVM as a Machine

**Wanted Properties**

## Current Efforts

Writing Specification

Testing the Specification

Proving Ethereum Contracts Correct

## Outlook

# Properties Wanted about a Contract


## Safety Properties

- ▶ only this kind of callers can alter storage
- ▶ only this kind of callers can decrease the balance<sup>3</sup>
- ▶ the invalid opcode `0xfd` is never hit  
(Some compilers encode safety properties using `0xfd`)

## Game Theoretic / Cryptographic Properties

“bidding honestly” should be a dominant strategy  
if a contract implements a second-price sealed auction correctly.

---

<sup>3</sup>Anyone can add balance to any account ☺  16/28



# Phases of EVM Modeling

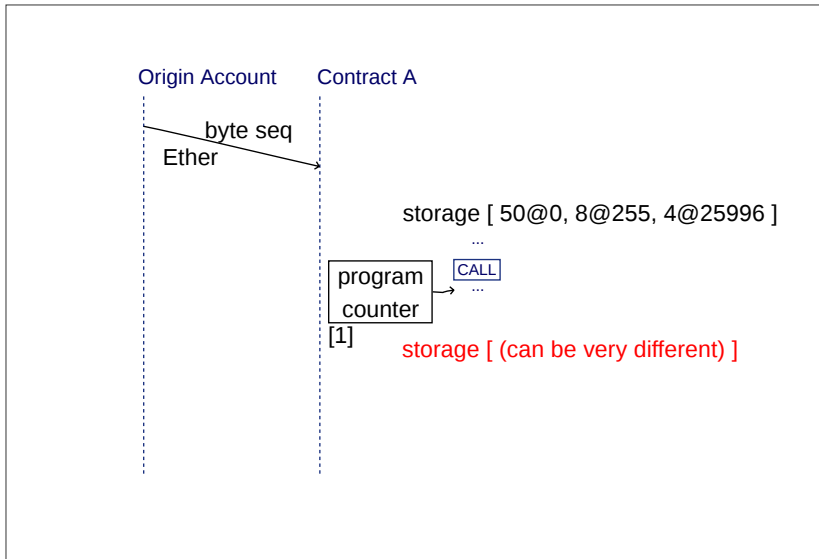
Phase 1 single call—done

Phase 2 caller-callee interaction—in testing & debugging

Phase 3 follow the blockchain—not started

# Phase 1: Take the Single Invocation's View

Involves some artificial nondeterminism.



# Special Treatment of CALL

During CALL instruction, nested calls can enter our program.

Our black box treatment of CALL during phase 1

- ▶ by default, the storage and the balance change arbitrarily during a CALL.
- ▶ optionally, you can impose an invariant of the contract, which is assumed to be kept during a CALL but you are supposed to prove the invariant.

# Outline

## Problem

Motivation

EVM as a Machine

Wanted Properties

## Current Efforts

Writing Specification

Testing the Specification

Proving Ethereum Contracts Correct

## Outlook

# Lem

- ▶ a specification language
- ▶ translates into HOL4, Isabelle/HOL, OCaml (and Coq)

## How I started using Lem

1. I started this project in 2015 in Coq.
2. I tried Isabelle/HOL and my proofs got shorter.
3. Sami Mäkelä saw this and started the Lem version.

# Outline

## Problem

Motivation

EVM as a Machine

Wanted Properties

## Current Efforts

Writing Specification

**Testing the Specification**

Proving Ethereum Contracts Correct

## Outlook

# OCaml for Testing

- ▶ Lem to OCaml extraction
- ▶ OCaml code to parse test cases (simplest “VMTest” format)
  
- ▶ Luckily, EVM has test suites
  - ▶ for implementations in Python, Go, Rust, C++, . . . need to match exactly
- ▶ VM Test suite: 40,617 cases (24 cases skipped; they involve multiple calls)

Need to run other formats.

# Outline

## Problem

Motivation

EVM as a Machine

Wanted Properties

## Current Efforts

Writing Specification

Testing the Specification

**Proving Ethereum Contracts Correct**

## Outlook



# Isabelle/HOL for Proving

Lem to Isabelle/HOL translation seems to be working.

## As an off-the-shelf symbolic executor

Keeping the input  $x$ , without making it concrete.

Just watching the states evolve after each instruction.

Soon we see one stack element

“ $\neg$  (the first four bytes of  $x == 0x44552211)$ ”

Number & size of the cases explode.

One instruction takes 15 seconds for a realistic code.

## Separation logic

Amani Sidney and Maksym Bortin ported a separation logic library onto EVM.

Compositional reasoning.

# Proving Theorems about Ethereum Programs in Isabelle/HOL

## With symbolic execution

One theorem about a program (501 instructions) says:

- ▶ If the caller's address is not at the storage index 1, the call cannot decrease the balance
- ▶ On the same condition, the call cannot change the storage

## With separation logic

I deployed a proven wallet as a bounty program (since closed).

# Way Ahead

## Ongoing

- ▶ testing the formalization of a whole transaction, containing transactions containing calls
- ▶ verified compiler for a simple language (by Sami Mäkelä)

## Not started

- ▶ implementing the next protocol change
- ▶ common Ethereum contract method/argument encoding
- ▶ connect to test/main network

## A Competitor

- ▶ KEVM by Grigore Rosu and his team: EVM definition in K-framework, gets some tools “for free”.

# Summary

- ▶ We defined EVM for proof assistants Isabelle/HOL, Coq and HOL4
  - ▶ The EVM definition is usable for proving Ethereum contracts against a specification
  - ▶ We found mistakes in the  $\text{\LaTeX}$  spec while writing and testing our definition.
- 
- ▶ Proof/tool/language/protocol developments in the proof assistants welcome  
<https://github.com/pirapira/eth-isabelle>  
(Apache License ver. 2 except material from Lem)