Validating Smart Contract Execution Across a Heterogenous Collection

A Proposal

Paddy Krishnan

Oracle Labs, Brisbane, Australia

July 2018

The following is intended to provide some insight into a line of research in Oracle Labs. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. Oracle reserves the right to alter its development plans and practices at any time, and the development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle. Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.

Agenda

- Background
- Challenges
- Proposal and Open Problems
- Q & A

Background

- **Distributed consensus not new**
  - **Fischer Lynch Paterson : 1985 JACM**
  - **Presence of malicious users: Byzantine Fault-tolerance**
- **Blockchain: A particular architecture for distributed consensus**
  - **Also handles malicious users**
- **Distributed protocols enforce some desired property**
- **Cryptographic proof of work**
  - **Dwork/Naor 1992**

Background

- **Bitcoin: Enforces application specific properties**
  - **Prevent double spending**
  - **Privacy preservation**
- **Ethereum**
  - **Transaction based state machine**
  - **GHOST protocol for consensus**
  - **Transitions between the states expressed as a program**
  - **EVM: Stack based virtual machine**
  - **Programming languages like Solidity**

Challenges

- **Scalability**
  - **Number of participants**
  - **Transactions costs**
  - **Size of block and maximum length of chain,**
- **Security assurance**
- **Heterogenous systems**
  - **Need single “truth”**
  - **Different implementations**

Challenges

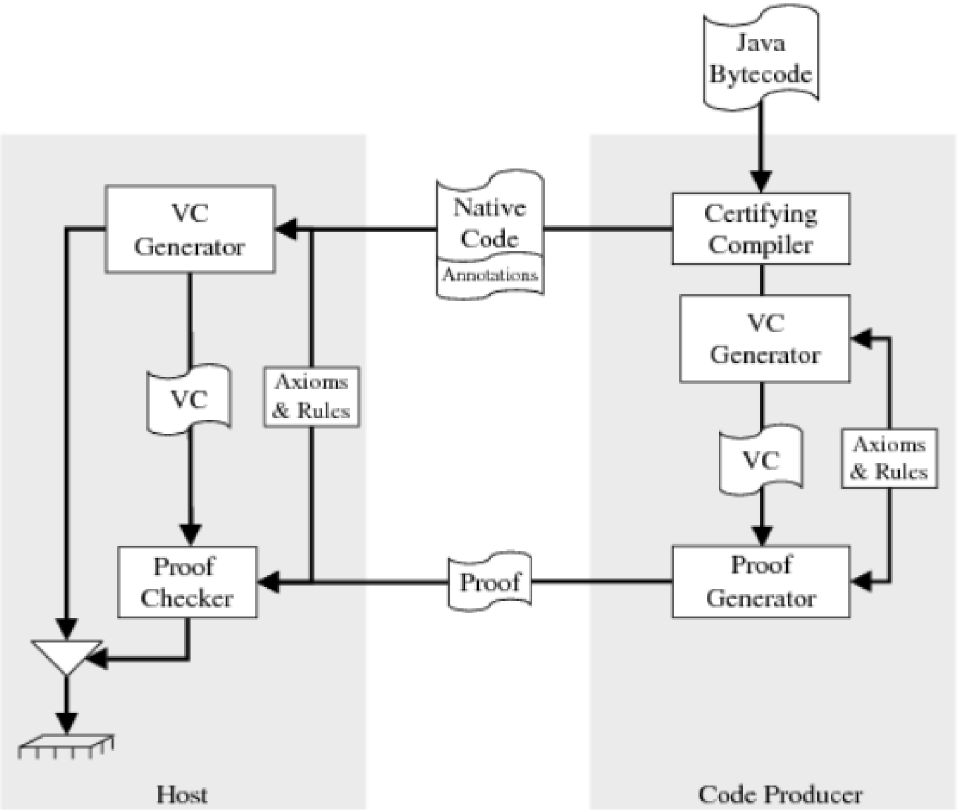
- **Is the smart contract correct?**
  - **Does it meet the desired properties**
  - **Very similar to program verification**
- **Are all the participants executing the same contract?**
  - **Is there any tampering?**
  - **Are integrity constraints satisfied?**
- **Is the downloaded code correct?**

Proposal

- **Can ideas such as**
  - **Proof carrying code or Proof carrying data be useful?**
  - **Proof carrying code was proposed in 1998 by Necula and Lee**
- **The key idea**
  - **Generator of new state generates a proof that the transition is valid**
  - **The user of the new state checks that the proof is valid before accepting state**
- **Premise**
  - **Generating proof is harder than verifying the proof**
  - **The onus is on generator to demonstrate “correctness”**

Proof Carrying Code

Architecture



Open Problems

- Can Turing complete languages be supported in practice?
- What are the logics to express the desired properties?
  - Do we need tools such as Coq or Isabelle?
- Are there decidable subsets where automated techniques can be used?
  - What are the tradeoffs between expressive power, security guarantees and applicability?
  - What is the tradeoff between the costs of proof generation and proof checking?

Open Problems

- What are the sizes of typical proofs?
  - Can the proof be converted to a ‘tactic’?
- Can they be stored on the chain?
  - If not do we need to use trusted entities?
  - What is the level of trust required?
- Can a class of proof-checkers be used?

Open Problems

- **Can such ideas be used with state channels?**
- **Do the proofs need to be history sensitive or do they apply only to transitions?**
- **Can other techniques such as probabilistic proof-systems or approximate algorithms suffice?**
  - **They may not give 100% guarantee but at an acceptable risk level**

Q&A

Integrated Cloud

Applications & Platform Services

ORACLE®