Formally verifying a zkEVM

May 2024 - For Polygon

Formal Land

Vitalik's goal for the year

Formal verification of zkEVMs, for the arithmetization

- Formal verification is checking a program correct for **all possible inputs**
- Requires dedicated techniques as **inputs are infinite**
- This is a **critical goal** for the Ethereum Foundation

Our past projects and Polygon

1. L1 of Tezos
2. coq-of-rust
3. coq-of-python
4. coq-of-solidity
5. Polygon zkEVM

L1 Tezos

- Formal verification of the core of Tezos
- 80% of files with some proofs
- Interpreter, storage, backward-compatibility
- https://formal-land.gitlab.io/coq-tezos-of-ocaml/

coq-of-rust

- Import almost any Rust code to the formal verification tool Coq
- https://github.com/formal-land/coq-of-rust

coq-of-python

- New project, ongoing
- For the EVM specification
- https://github.com/formal-land/coq-of-python
- Combined with coq-of-rust to verify the Revm version of the EVM

coq-of-solidity

- Just starting
- A formal verification tool for Solidity with Coq
- Reusing the same techniques as coq-of-rust

Verifying Polygon's zkEVM

Our proposition

1

FORMALIZE THE CODE

- Import zkEVM to Coq with coq-of-rust
- Process the output so that it is suitable for formal verification

2

SHOW SOUNDNESS

- Verify the arithmetization of all the operations
- Show that the behavior is the same as in the reference EVM, for all possible inputs

Thanks