Formally verifying a zkEVM

May 2024 - For Polygon



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

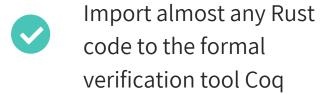
- 1 L1 of Tezos
- 2 coq-of-rust
- 3 coq-of-python
- 4 coq-of-solidity
- 5 Polygon zkEVM

Our past projects and Polygon



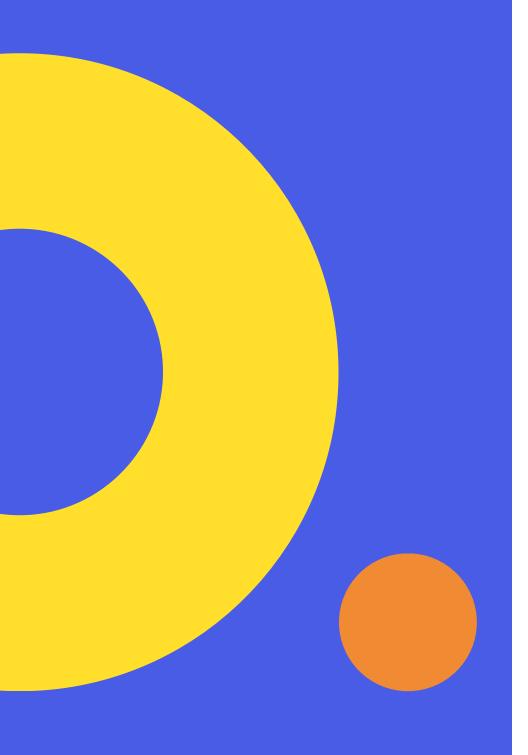
- Formal verification of the core of Tezos
- 80% of files with some proofs
- Interpreter, storage, backwardcompatibility
- https://formal-land.gitlab.io/coq-tezos-of-ocaml/

coq-of-rust





https://github.com/formalland/coq-of-rust



coq-of-python

- New project, ongoing
- For the EVM specification
- https://github.com/formalland/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



FORMALIZE THE CODE

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



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