Functional Verification of Tezos Smart Contracts in Coq

Raphaël Cauderlier

Journée scientifique Inria - Nomadic Labs

Paris, September 21, 2020

Introduction

Motivation

Smart contracts are the perfect setup for formal methods:

- bounded complexity
- hard to update once deployed
- manage valuable assets

Mi-Cho-Coq

Mi-Cho-Coq

Deep embedding in Coq of the Michelson smart-contract language

- lexer, Menhir parser, macro expander, type checker, fuel-based evaluator, pretty-printer
- Coq proof of type-checker correctness
- automated tests of compliance with the Michelson reference implementation
- weakest-precondition calculus for functional verification of smart contracts

Verified smart contracts

- manager: Tezos' default smart contract
- multisig: advanced authentication and loops
- spending limit: reasoning on time, encoding of a queue
- Dexter decentralized exchange (ongoing): complex contract

Albert

The Albert intermediate language

Goals:

- common suffix of most compilers to Michelson
- optimizing
- certified

Choices:

- abstract the stack
- linear type system
- some easy features: inlined functions and n-ary records and variants

Language specification

- syntax, typing, and semantics specified using Ott
- modular specification (one file per language construction)
- from one source
  - Menhir parser
  - LaTeX documentation
  - Coq AST, typing, and semantic relations
- Coq proofs of subject reduction and progress on the core fragment

Compiler

- from the generated Coq AST to Mi-Cho-Coq
- written in Coq, but not yet certified
  - coverage testing using `bisect_ppx`
  - Coq proofs for simple optimisations at the Michelson level

Conclusion

Conclusion

- Michelson is formalized in Coq
- Applications
  - Verification of some interesting smart-contracts
  - Certified compilation.

Ongoing and Future Work

- on Albert
  - prove meta theory for the full language
  - improve and certify the compiler
- on Mi-Cho-Coq
  - prove Mi-Cho-Coq $\Leftrightarrow$ Michelson reference implementation (using coq-of-ocaml)
  - go beyond functional verification: cost model, contract life, mutual and recursive calls

Links

- Michelson documentation:
  - `https://tezos.gitlab.io/whitedoc/michelson.html`
  - `https://michelson.nomadic-labs.com`
- code repositories and articles:
  - `https://nomadic-labs.com/software_verification.html`