Overview

This document describes the formalization in the Rocq prover for the following article.
Anonymous: Strategy-aware Liquidity for Account-based Blockchains. (under submission)

The formalization as contained in this folder is currently for the assessment of the above article. Please do not distribute the formalization or use it for other purposes.

The formalization is based on the ConCert framework:
GitHub - AU-COBRA/ConCert: A framework for smart contract verification in Coq

The formalization mainly includes:

1. Model of transaction execution as driven by the strategies of the honest users and the adversary, based on the model of smart contract execution in ConCert
2. Formalization of the strategy-aware liquidity property in the article, and the meta-theoretical properties of the liquidity property (incl. monotonicity and connection with strategy-less liquidity)
3. A proof method for the verification of strategy-aware liquidity, together with the soundness proof of the method
4. A proof method for the refutation of strategy-aware liquidity, together with the soundness proof of the method
5. Formalization of the smart contracts c_{fm}, $c_{fm}^{\times}$, the escrow contract, and the honeypot contract, accompanied with the proofs for verifying or refuting strategy-aware liquidity for these smart contracts

Paper-to-code Correspondence

The following table shows the correspondence between contents in the article and their mechanized counterpart in the Rocq prover.

Part in Manuscript	Definition/Result	File in Mechanization	Entry
Section 3.1	$\sigma \xrightarrow{\beta} \sigma'$	strat_liquidity/StratModel.v	`Definition transition`
Section 3.1	*init-ctr*	strat_liquidity/StratModel.v	`Definition is_init_state`
Section 3.2	*Definition 1 (strategies)*	strat_liquidity/StratModel.v	`Definition strat`
Section 3.2	$\delta^{as} \vdash \pi \to \pi'$	strat_liquidity/StratModel.v	`Definition stratDrive`
Section 4	$(\delta_{\mathrm{u}}^{as1}, \delta_{\mathrm{a}}^{as2}) \vdash \pi \rightsquigarrow \pi'$	strat_liquidity/StratModel.v	`Inductive interleavedExecution`
Section 4	*usl*, *asl*	strat_liquidity/StratModel.v	`Inductive userLiquidates …` `with envProgress …`
Section 4	*Definition 2 (strategy-aware liquidity)*	strat_liquidity/StratModel.v	`Definition strategy_aware_liquidity`
Section 5.1	*Definition 3 (Relative strength of strategies)*	strat_liquidity/Mono.v	`Definition strat_subset`
Section 5.1	*Theorem 1 (monotonicity)*	strat_liquidity/Mono.v	`Theorem strat_liquidity_mono_wrt_env`
Section 5.2	*Definition 4 (Basic liquidity)*	strat_liquidity/StratModel.v	`Definition basic_liquidity`
Section 5.2	*Theorem 2*	strat_liquidity/Connection.v	`Theorem SL_equiv_BL_with_empty_env_`

	(connection between liquidity notions)		`and_complete_user`
Section 6.1	*Theorem 3 (soundness of verification)*	strat_liquidity/ProofLib.v	`Theorem soundness`
Section 6.2	*Theorem 4 (soundness of refutation)*	strat_liquidity/ProofLib.v	`Theorem rft_soundness`
Section 7.2	*Theorem 5 (verification result for* c_{fm}*)*	strat_liquidity/examples/ fundsManagement/ fdmgmt_sat_liq.v	`Theorem fm_sat_strat_liquidity`
Section 7.2	*Theorem 6 (refutation result for* $c_{\text{fm}}^{\times}$*)*	strat_liquidity/examples/ fundsManagement/ fdmgmt_nsat_liq.v	`Theorem fm_unsat_strat_liquidity`
Section 7.2	*Verification result for escrow contract*	strat_liquidity/examples/ escrow/escrow_sat_liq.v	`Theorem escrow_sat_liq_honest_buyer` `Theorem escrow_sat_liq_honest_seller`
Section 7.2	*Refutation result for honeypot contract*	strat_liquidity/examples/ honeypot/honeypot_nsat_liq.v	`Theorem honeypot_unsat_strat_liquidity`

Building and Running the Proof Scripts

This formalization is developed using Rocq (Coq) 8.16.1

To build the proof scripts, first ensure that the “bin” folder of the Coq installation is in the system path.

A makefile should be generated from _CoqProject with the following command:

```
coq_makefile -f _CoqProject -o Makefile
```

To build the development, issue the "`make`" command in the top-level folder.

After the proof scripts are built, they can be run using Proof General or CoqIDE.