Equilibria in DeFi from State Context Inspection

James Hsin-yu Chiang[1], Conor McMenamin[2], and Margherita Renieri (Speaker)[3]

[1] *Technical University of Denmark, Lyngby, Denmark*
[2] *Universitat Pompeu Fabra, Barcelona, Spain*
[3] *IMT School for Advanced Studies Lucca, Lucca, Italy*

Abstract

In the design of decentralized finance (DeFi) [WPG+21], a desirable equilibrium for protocols is to ensure block producers are indifferent to interacting with the protocol. In isolation, an automatic market maker (AMM) exposing no arbitrage opportunities represents such a desirable, well-priced state; it can be seen as a fair reflection of the true price of the AMM tokens. This motivates attempts to construct price oracles from AMMs in such a state. Unfortunately, composition of DeFi applications can result in unintended incentives. For example, the rational strategy may be to manipulate oracles beyond these fair prices in order to trigger profitable liquidations from a composed DeFi lending protocol [QZLG21]; formal frameworks [BDKJ23, BZ23] have been introduced to analyze such mal-incentives emerging from DeFi composition. In this work, we investigate the problem of achieving desirable equilibria states (e.g. fair pricing in AMM's) under DeFi protocol composition by introducing notions of *state context* which expose execution traces. These can be inspected by DeFi smart contracts at run-time, providing context for protocols beyond basic user inputs. We examine whether intuitive notions of protocol equilibria can be encoded as *policies* enforceable on state contexts and execution traces.

Keywords— Decentralized Finance, Contract Composition, Incentives

1 Talk Proposal

We first introduce a definition of a *Gobal State Context*; informally, it captures the execution of the blockchain state machine up to the evaluation of a specific transaction txId; for simplicity, let txId denote the full transaction body required for evaluation by the smart contract virtual machine.

Definition 1 (**Global state context of depth** n)**.** *Let* txId *be evaluated on blockchain state* Γ *at block height* B*. Further, let* $\Gamma_0 \rightarrow^{\lambda} \Gamma$ *denote the execution of the blockchain state machine preceeding state* Γ*. The global state context (of depth* n*) of* txId *evaluated on* Γ *is given by*

$$\mathsf{ctx}_n\left(\Gamma, \mathsf{txId}\right) = \Gamma' \xrightarrow{\lambda'} \Gamma \xrightarrow{B:\mathsf{txId}} = \Gamma' \xrightarrow{B_1:\mathsf{txId}_1} \xrightarrow{B_2:\mathsf{txId}_2} \cdots \xrightarrow{B_n:\mathsf{txId}_n} \Gamma \xrightarrow{B:\mathsf{txId}}$$

where λ' *is obtained by removing a prefix from* λ *and all transactions in* λ' *are located in the current and* n *proceeding blocks; that is* $B_1 \geq B - n$*.*

We further define a *Local State Context* which permits the inspection of *inter-contract* calls during the execution of the current transaction txId. We presume an EVM[1]-like state machine, hosting object-like contracts exposing function interfaces callable by users and external contracts.

Definition 2 (**Local state context**)**.** *Let* txId *be the transaction evaluated on blockchain state* Γ*, e.g.* $\Gamma \rightarrow^{\mathsf{txId}}$*. Then, let a* j*-length prefix of the full inter-contract call sequence induced by the evaluation of* txId *on* Γ *be given by*

$$\mathsf{ctx}\left(\Gamma, \mathsf{txId}, j\right) = \Gamma \xrightarrow{C_1:\mathsf{fn}_1} \xrightarrow{C_2:\mathsf{fn}_2} \cdots \xrightarrow{C_j:\mathsf{fn}_j}$$

[1] Ethereum Virtual Machine

where $C_1 : \mathsf{fn}_1$ denotes the initial contract function called by txId, and $C_j : \mathsf{fn}_j$ be the j'th contract call under evaluation. We define both Γ and the j-length prefix of the call sequence induced by evaluating txid on Γ as the local state context of contract function $C : \mathsf{fn}_j$.

Next, we permit contract functions to *inspect* the *Global* and *Local State Context*s and *enforce* policies upon their evaluation. Concretely,

Definition 3 (**Context policy**). *Let a contract function $C : \mathsf{fn}$ implement a context predicate $\mathcal{P}^{ctx}$. Upon receiving a message call, $C : \mathsf{fn}$ will continue execution iff both local and global state context of depth n satisfy $\mathcal{P}^{ctx}$.*

We are particularly interested in how context policies can contribute to a blockchain ecosystem where individual components are more aware of the mal-incentives caused by potential dependencies on external component states. We provide the following example of mal-incentives resulting from composition, closely resembling the real-world price oracle attack analyzed in [QZLG21], and sketch possible mitigations enabled by context policies.

Example 1. *Consider a blockchain in a well-priced state, to which a player adds a liquidatable position. This position liquidates if the price of the AMM being used as the oracle drops 1% from the current well-priced state. Dropping the price of the oracle by trading with the AMM (AMM:swap) results in a loss of \$100 for the block producer, but buying the resultant liquidated collateral has an expected profit of \$200 (Coll:liquidate). As such, a rational block producer will always move the price of the oracle to extract the profit from the liquidation.*

$$ctx_1\,(\Gamma', \mathsf{txId}_2) = \Gamma \xrightarrow{B:\mathsf{txId}_1} \Gamma' \xrightarrow{B:\mathsf{txId}_2} = \Gamma \xrightarrow{AMM:swap} \Gamma' \xrightarrow{Coll:liquidate}$$

In contrast, an ideal price oracle requires arbitrageurs to continuously extract all AMM arbitrage and ignore all other incentives, which would prevent liquidations to be triggered by the aforementioned oracle attack; however, in the presence of the a block producer controlling the ordering of interactions with the AMM, such an ideal price oracle clearly cannot be realized, as the block producers will consider all incentives exposed by DeFi protocol composition.

Our proposed context policies (Definition 3) can specify desired/undesired state transitions. A desirable goal for an AMM-based price oracle is to construct some notion of a well-priced price oracle for triggering liquidations from public AMM states. A possible unacceptable context for an oracle based on AMM prices is a large mid-block price deviation far from the last well-priced state (final state of previous block for example). Let txId_1 below denote the last trade on the AMM in the block B_1 proceeding current block B_2 where the liquidation occurs; the oracle logic may impose a maximum permitted distance on the AMM price between Γ'' in B_2 and Γ' in B_1 via its state context predicate.

$$\Gamma \xrightarrow{B_1:\mathsf{txId}_1} \Gamma' \cdots \Gamma'' \xrightarrow{B_2:\mathsf{txId}_2} = \Gamma \xrightarrow{\mathsf{AMM:swap}} \Gamma' \cdots \Gamma'' \xrightarrow{\mathsf{Coll:liquidate}}$$

Should this predicate not be satisfied, then the liquidation will not be possible on state Γ'', protecting against price manipulations occurring in the same block. Other context policies defending against this include adjusting the oracle to return time-/volume-weighted average prices over multiple blocks rather than the current implied price, restricting the oracle to AMMs that settle all orders in a block at a single price, and/or auctioning off the collateral over multiple blocks, and reducing the incentive to manipulate the liquidation oracle for any individual block producer. All of these choices have trade-offs for the liquidation protocol, but provide motivation to investigate the capabilities and limitations of context policies.

If accepted, our talk will motivate and introduce the formal framework of state context and policies. We specify formal properties of context policies for common protocol compositions in Decentralized Finance and analyze their effectiveness in reducing mal-incentives occuring in price oracles [QZLG21], wash-trading [WWL+21, vWJRR22] and governance voting [GPH+20, HN22], which are amplified by flash-loans [WWL+21]. Finally, we discuss the practicalities of natively adopting state context inspection functionality in an EVM-like implementation.

References

[BDKJ23] K. Babel, P. Daian, M. Kelkar, and A. Juels. Clockwork finance: Automated analysis of economic security in smart contracts. In *2023 IEEE Symposium on Security and Privacy (SP)*, 2023. https://doi.ieeecomputersociety.org/10.1109/SP46215.2023.00036.

[BZ23] Massimo Bartoletti and Roberto Zunino. A theoretical basis for blockchain extractable value. *arXiv preprint arXiv:2302.02154*, 2023. https://arxiv.org/abs/2302.02154.

[GPH+20] Lewis Gudgeon, Daniel Perez, Dominik Harz, Benjamin Livshits, and Arthur Gervais. The decentralized financial crisis. In *2020 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 1–15. IEEE, 2020. https://doi.org/10.48550/arXiv.2002.08099.

[HN22] Khang Hoang and Giap Nguyen. How to stole an election: Beanstalk dao $80 million flashloan attack study case, 2022. https://blog.verichains.io/p/how-to-stole-an-election-beanstalk.

[QZLG21] Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. Attacking the defi ecosystem with flash loans for fun and profit. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part I*, pages 3–32. Springer, 2021. https://doi.org/10.1007/978-3-662-64322-8_1.

[vWJRR22] Victor von Wachter, Johannes Rude Jensen, Ferdinand Regner, and Omri Ross. Nft wash trading: Quantifying suspicious behaviour in nft markets. *arXiv preprint arXiv:2202.03866*, 2022. https://arxiv.org/abs/2202.03866.

[WPG+21] Sam M Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J Knottenbelt. Sok: Decentralized finance (defi). *arXiv e-prints*, pages arXiv–2101, 2021. https://doi.org/10.48550/arXiv.2101.08778.

[WWL+21] Dabao Wang, Siwei Wu, Ziling Lin, Lei Wu, Xingliang Yuan, Yajin Zhou, Haoyu Wang, and Kui Ren. Towards a first step to understand flash loan and its applications in defi ecosystem. In *Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing*, SBC '21, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3457977.3460301.