The “offer-is-code” approach to Decentralised Exchanges

The Mangrove team 🌴🌴🌴

v5 revised Nov 2021

This paper introduces a novel and quite general approach to trading which we term “offer-is-code”. We demonstrate the idea by building around it the Mangrove protocol, a decentralised spot market. Mangrove offers are sorted in an order book. Each offer contains a callback address which the matching engine calls when the offer is effectively matched. Incentives are set to make sure that offers are not empty promises.

The offer-is-code approach opens up new strategies for liquidity providers (LPs).

1. (last looks) LPs can incorporate defensive code to cancel offers if market conditions are unfavourable, and avoid price exposure leading to costly arbitrage.
2. (reactive liquidity) LPs can post offers that are not fully provisioned. It is enough that their code brings the promised liquidity at match-time. In the meantime, it can be put to work.
3. (persistence) LPs can redisplay atomically their liquidity after a match, without latency, in the fashion of automated market makers.

1 Goodbye TVL

Marketplaces foster the liquidity of financial assets by bringing together a large number of sellers and buyers. Their orders are collected and then matched so as to maximise the gains from trade. This matching process would be of little value if traders were able to renege on their commitments to trade at a given price. The traditional finance (TradFi) solution is to rely on third parties that take temporary custody of the assets and ensure that promises are fulfilled [6]. Decentralised finance (DeFi) dispenses with intermediates by settling trades on a blockchain. DeFi applications leverage the commitment power of smart contracts to enable financial transactions that do not require traders to relinquish control over their assets. Assets

traded through DeFi protocols cannot be seized and are immune from the counterparty risk of intermediaries.

DeFi is a nascent technology whose earliest applications have followed the modus operandi of centralised marketplaces by monitoring and settling limit orders on-chain. However, it has become clear that the fundamental benefits of DeFi have to be balanced against the operational costs (also known as gas costs) of processing transactions in a decentralised environment. The difficulty to efficiently handle limit order books has prompted the arrival of leaner protocols. Most notably, automated market makers (AMM), such as Uniswap v2 [1], have established themselves as the dominant exchange protocols.

The main advantage of AMMs is that they rely on simple rules to compute the price of a trade, and redisplay their liquidity pools after the execution of a trade. But there are problems:

1. (no information) The pricing schedules of AMMs lack the informational content of limit-order books, a crudeness that casts doubt on the capacity of AMMs to attract sophisticated liquidity providers, and which concretely leads to poor returns for LPs who must share their revenues with arbitrageurs.
2. (fragmentation) AMMs lock assets in liquidity pools. For each combination of assets to be swapped, a specific liquidity pool has to be created. The more trading pairs get listed over time, the more fragmented the DeFi ecosystem becomes. This leads to markets with low liquidity.
3. (dormant assets) Capital locked into pools is under-utilised, leading to substantial capital inefficiencies.

The Mangrove protocol addresses the above DeFi pain points. First, Mangrove matching and price-formation processes rely on order books which takes care of the first point (no information). The advent of cheaper layer 2 substrates (such as Polygon, Arbitrum, and soon StarkNet) makes the prospect of deploying an order book less daunting. Second, and this is the key, Mangrove books let liquidity providers place offers which are themselves code. A rather natural idea in a world of smart contracts. In particular, an offer does not need to lock up the promised tokens upfront. Instead, an offer comes equipped with algorithmic recipes to:

1. (sourcing) obtain the assets when the offer is matched, and
2. (persistence) repost itself on the book.

Thanks to the sourcing recipe, liquidity can be shared, borrowed, or lent, and yet, at the same time, be displayed in a Mangrove book ready for trade. There are no restrictions on the types of offers that can be posted. Concretely, and provided they write the connectors, LPs can source their liquidity from arbitrary contracts. The second point (fragmentation) becomes less of an issue. And of course the third point (dormant assets) is solved.

Mangrove locks no value. The power of the offer-is-code approach is that it makes Mangrove a non-rival conduit for one's liquidity. This seems ground enough to raise interest in the protocol, and a strong point for adoption. There are also specific novel opportunities that follow from the initial idea. For instance, not only can participants bring together virtually their hitherto scattered liquidity, but they can also publish that liquidity multiply. Also, thanks to the persistence recipe, LPs can save on maintaining an off-chain bot, and use their on-chain code to repost offers with zero-latency. We elaborate on these novel uses case below (§3).

The rest of the paper is structured as follows: first, we describe the protocol and incentives (§2); then we propose use cases for liquidity providers (§3); finally, we discuss briefly the protocol's governance and possible paths to adoption (§4).

2 The Protocol

Mangrove is permissionless and exclusively on-chain: any user or contract can post an offer signalling their willingness to sell up to a given quantity of assets at a given price. In addition, the protocol asks providers to place offers which contain the code to be executed (on-chain) at match-time.

Specifically, a Mangrove offer consists in:

- a pair (x, p) representing a promise to trade up to an amount x of a particular asset at price p,
- an upper bound on the amount of gas requested by the offer to run its course,
- a provision in the currency native to the substrate chain (ETH on Ethereum, Matic on Polygon, etc), and
- a callback address which the matching engine uses to call on the provider's code when the offer is matched, and later at repost-time.

Gas costs are evaluated at a gas price which is dynamically updated by the protocol. Offers whose provision does not cover their gas costs (evaluated at the protocol's gas price) are rejected (whether submitted de novo or re-submitted via the persistence mechanism).

There is no restriction on the code that can be attached to an offer. A controlled form of reentrancy allows offers to persist in the book and redisplay their liquidity after a match (Fig. 2). Persistence allows providers to run across the Mangrove books what amounts to a private AMM tailored to their on-chain liquidity pools. These private AMMs can be dynamic and listen to their own signal chain (on- and off-chain). TradFi traders commonly fantasise about running their strategies from within the exchange. In a way, this what the Mangrove protocol allows them to do. (Incidentally, the "offer-is-code" approach may be of interest to TradFi as well.)

Offers are triggered when matched with an incoming market order. As originators of the transaction, liquidity takers have to pay the gas costs requested by the offers their order is matched with. Should one of the matched offers fail, its provision is drawn upon in order to reimburse the taker as follows:

- the matching engine reverts the modifications incurred by the execution of the failed offer
- the taker is returned her initial payment drawing from the provider's provision, and
- the engine proceeds to the next offer.

The only drawback of fails from the taker's standpoint is that they may exhaust the amount of gas she initially funded for the completion of her market order. To avoid this, Mangrove has two counter-measures in place. First, it simply supplies takers with an estimate of the gas needed for the execution of their market order assuming no offer fails. And takers are advised to over-fund their transaction to ensure that their order can weather a few fails. Second the protocol incentivises external agents to keep the books clean by requiring that the provision attached to an offer slightly exceeds the cost to run it. When an offer is seen to fail (e.g. by dry-running it), keepers can snipe it, seize the bounty, and rid the book of it.

Compensations for fails are necessary because Mangrove does not enforce any gatekeeping. Nothing prevents LPs from posting offers that will always fail. To ensure that the offers displayed on the book are credible, it must be costly for providers to post orders that are bound to fail. Hence the security of the order book is guaranteed by the provisions of providers. This design results from the structure of the incentive problem: providers being the source of moral hazard, they have to advance funds that can be seized to punish those that try to clog the order book with failing orders.

One can contrast this policy with that of most current DEXs to ensure the reliability of liquidity sources. DEXs commonly treat this requirement for liquidity as a hard constraint. Mangrove's bond-and-slash scheme meshes with the fact that its order books list promises to provide liquidity instead of actual commitments (for all the reasons discussed in §1).

Another equally important aspect of the market microstructure is to ensure a minimum size and density in offers. Density measures the ratio of traded volume to gas costs for a given offer. When density is low, a large share of the trade is dissipated in execution costs. Setting a lower bound on density therefore ensures that the book is not polluted by (possibly maliciously) inefficient offers.

Fig. 1 summarises the interactions between the Mangrove DEX and external contracts. Fig. 2 details the execution flows under the maker-driven mode (see §3 for uses of the taker-driven mode).

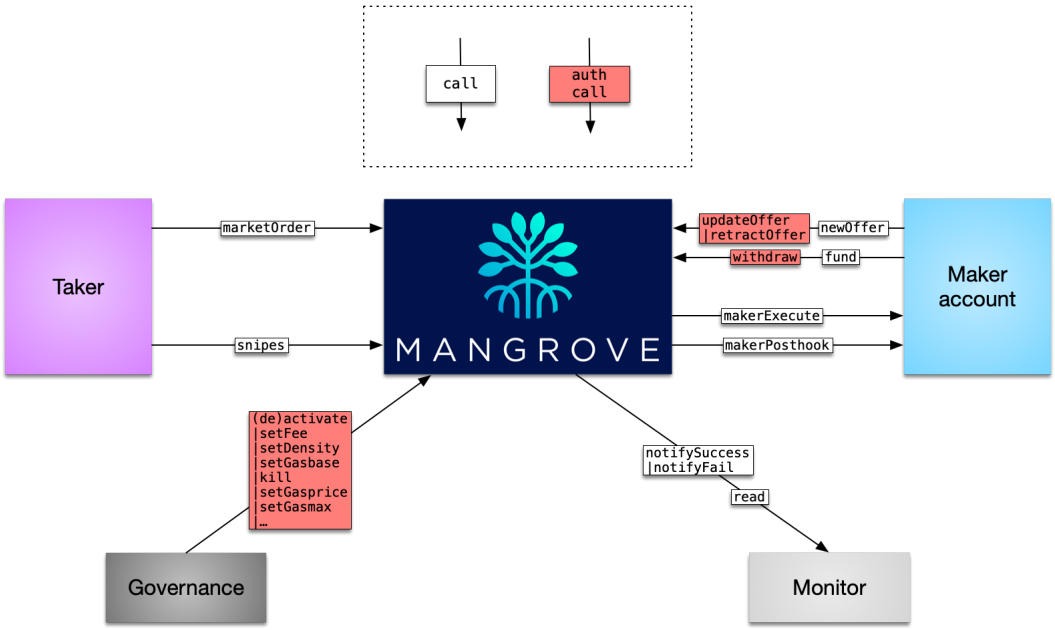

Figure 1: Boxes are contracts and arrows are function calls. Takers consume offers through market orders (`marketOrder`) or by sniping specific offers (`snipes`). Liquidity providers create (`newOffer`), update (`updateOffer`), and cancel (`retractOffer`) offers using the Offer functions. They manage their provisions with fund (`fund`) and withdraw (`withdraw`). Depending on the selected mode of interaction (here maker-driven), Providers or Takers may receive callbacks (`makerTrade`, `makerPosthook`) during the execution of an order.

3 Use cases

We now present a series of use cases to illustrate the versatility of Mangrove, and then benefits it can bring to liquidity providers (hence indirectly to takers).

Clearly the fact that the promised assets are not required upfront by the Mangrove DEX opens up interesting strategies for providers to manage their liquidity. First, the promised assets can be put to work in liquidity pools from the local DeFi system. E.g., they can be loaned out on money markets, and earn passive returns, while waiting to be exchanged. Second, the funds flash-loaned from the taker can be used as collateral to borrow the promised assets. This gives us our first two use cases:

1. **Reactive liquidity (Lend):** The provider relies on a lending protocol (e.g. Compound [5]) to loan out the assets promised in Mangrove. Upon being called, the smart contract redeems the provider's claims and transfers the assets to the taker.

   (a) Economic Benefits: Provider does not forsake the returns on his assets

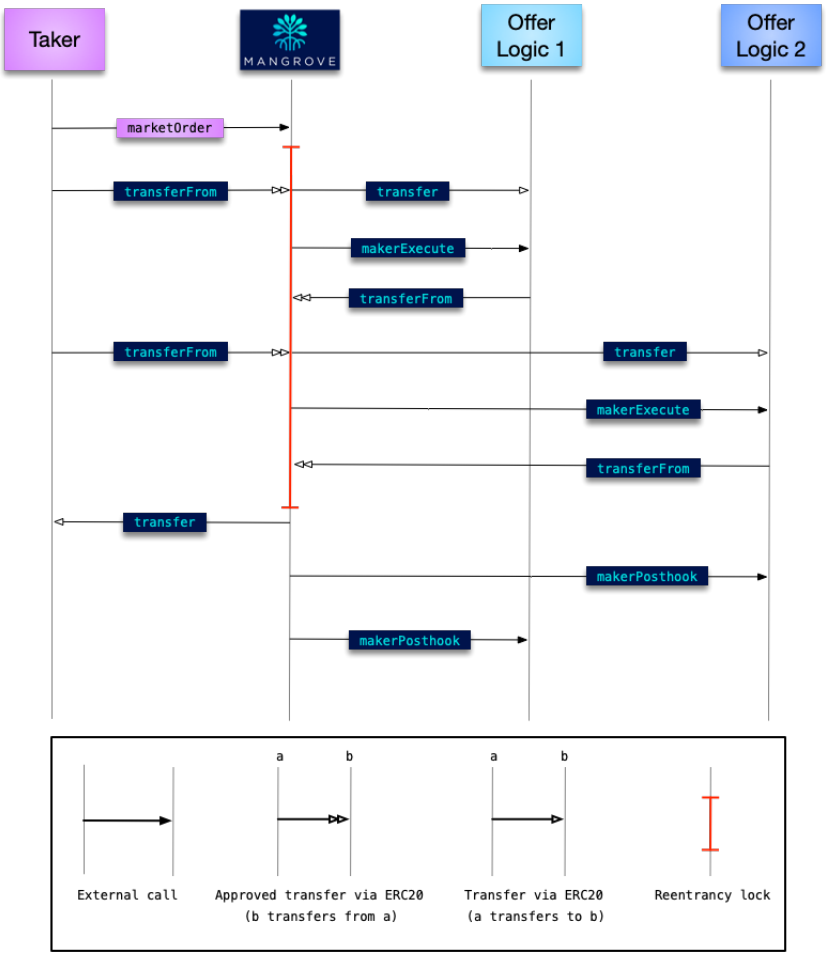

Figure 2: Sequence diagrams for the maker-driven interaction mode. The sequence starts with a market order where Offer1 is executed first, and then Offer2. In this mode, Taker flash-loans the appropriate amount of B to each Offer, and expects the correct amount of A in return.

while its offer sits on the order book.

(b) Risk: Limited to the illiquidity risk of the lending protocol.

(c) Comparative Advantage: Economic benefits can be emulated within a standard DEX by offering to exchange the loan claims instead of the underlying assets (for instance by opening an order book for Compound's cDAIs instead of DAIs). With this approach, however, the burden of converting the claims is shifted towards takers. More problematically, it requires to create a specific order book for each lending protocol, thus importing their fragmentation within the structure of the DEX, whereas Mangrove gathers all the offers in a single order book as long as they

concern the same asset pair.

2. **Reactive liquidity (Borrow):** The provider opens a credit line with a lending protocol. When the offer's code is called, it merges the funds of the taker with those of the provider to borrow the promised assets through the lending protocol (assuming maker-driven mode, Fig. 2).

   (a) Economic Benefits: This strategy emulates the common use of credit lines between liquidity providers and centralised exchanges. It lowers the opportunity costs of providers by minimizing the amount of collateral that they have to own.

   (b) Risk: Besides the counterparty risk of the lending protocol, the order may fail if the borrowing rights are insufficient, for instance following a sudden increase in the price of the promised asset.

3. **Last look:** Include the verification of an oracle within the execution of the smart contract which cancels the order if the price-feed exceeds a given threshold.

   (a) Economic Benefits: This option protects the provider against the risk of front-running, for example when some takers attempt to snipe a highly profitable order before its cancellation has been processed [3]. Given the asynchronous nature of Blockchains and the risk of collusion between miners and takers, concerns about front-running are even more prevalent in decentralised environments than in centralised ones [4].

   (b) Risk: Manipulation of the oracle.

   (c) Note: failing early is advantageous as the bounty paid out for reneging is proportional to the actual gas costs; whether the offer stays on the book with a higher ask price or disappears is set in the persistence part of the code which should be made aware of the prior decision to fail.

4. **Market for flash-loans:** Create an order book A/A in taker-driven mode. A dual form of the basic contract allows one to flash-loan the liquidity of the provider to the taker, creating a competitive market for flash-loans.

   (a) Economic benefits: For takers, this unified source of flash-loans offers volume and better fees; for providers, it yields riskless instantaneous returns when their offers are scooped up.

   (b) Risk: No risk other than the usual fail risk at execution time.

   (c) Comparative advantage: At the moment, flash-loanable liquidity is scattered and accessible through ad hoc methods and priced un-competitively (with the notable exception of DAIs). Mangrove's market for flash-loans flexibly aggregates these sources of liquidity and put them on a level playing field where they compete on flash-fees.

5. **Leveraged liquidity:** Display shared liquidity on a set of pairs -e.g. an AB, BC, CA triangle in a way that the some of the volumes on offer exceeds one's capital. When liquidity is taken on one pair, volume-adjust the offers on the other markets to keep the strategy solvable. Notice that one of the pairs can be an ordinary (locking) DEX as long as liquidities can be atomically redeemed.
   (a) Economic benefits: liquidity providers can increase their turn-over.
   (b) Risk: No risk as the volume-adjustment is atomic with the execution of the one offer taken first.
   (c) Note: the less risk-averse strategist may let the illiquid remaining offers fail (early) as it may be that they are made liquid again before getting hit.
   (d) Note: Using the leveraged liquidity idea, active strategies -say for Uniswap v3 [2], can be duplicated on Mangrove with no additional capital.

There are more use cases. The Mangrove books can also be used as price-sensors. The trigger now is information (which the volume offered pays for) and the code offer can be seen as an on-chain bot enabling complex trades. With price sensors in mind, it is easy to build robust stop-loss orders with no off-chain components; hence requiring no off-chain infrastructure, and having zero-latency. Likewise, automated top-up bots can be set-up to rescue positions near liquidation. One can also use these signals to drive persistent strategies on AMMs. Given the constant evolution of the DeFi landscape, it is clear that our series of use cases only scratches the surface of the engineering space opened up by Mangrove.

4 Governance

The Mangrove protocol, properly implemented and deployed is a flexible order book generator that may be combined via the offer-as-code paradigm with other protocols to support sophisticated DeFi applications. It is best released as an open protocol the growth of which relies on engaging a community of developers and participants. The Mangrove core contracts should be distributed under such licences (e.g. the AGPL3 licence) which will allow the DeFi community to leverage the Mangrove infrastructure. In this way, one may hope to set in motion a decentralised and community-driven effort to build new libraries of reactive strategies, which will explore efficiently the Mangrove strategic design space.

The matching engine should be gas-optimised for the EVM (the abstract machine underpinning the execution of smart contracts) to keep operational costs at the lowest. To bring the full benefits of the protocol to bear, one will nevertheless have to deploy it on cheaper EVM-based chains such as Polygon, or Arbitrum. Again a community-driven effort will naturally home in on the substrate blockchains most apt to the task.

Within the remit of governance are the many parameters tuning the market microstructure, such as price ticks, minimum order size, lower bound on the density, etc. Beyond that, the most important mandate of the governance is the definition of the price structure of the platform. For instance, the protocol could ask liquidity providers to transfer for free a share of their provisions. Part of these fees could then be passed on to liquidity takers in order to incentivise their participation. Obviously, if makers are more difficult to attract, a reverse mechanism should be put in place wherein fees are transferred from liquidity takers to liquidity providers. As usual, deciding which side of the market is cross-subsidised will be based on the relative elasticity of their demands. The optimal balance may vary over time and the flexibility of Mangrove's governance will enable its community to adjust the price structure so as to respond to changes in market conditions. The fee structure is one instrument that can be used to implement the overall price structure which maximises adoption. Combined with generic DeFi tools to build adoption, it should provide the collective governance with flexible means for the optimisation and growth of the protocol.

References

[1] Hayden Adams, Noah Zinsmeister, and Dan Robinson. Uniswap v2 core, March 2020.

[2] Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. Uniswap v3 core, March 2021.

[3] Eric Budish, Peter Cramton, and John Shim. The high-frequency trading arms race: Frequent batch auctions as a market design response. *The Quarterly Journal of Economics*, 130(4):1547–1621, July 2015.

[4] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges, April 2019.

[5] Robert Leshner and Geoffrey Hayes. Compound: The money market protocol. Technical report, Compound Finance, Tech. Rep, February 2019.

[6] Ed Nosal and Robert Steigerwald. What is clearing and why is it important? *Chicago Fed Letter*, September 2010.