Context and Ideas for a new DEX on Cardano

2023-08-20

Matthieu Pizenberg
matthieu.pizenberg@gmail.com

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit.

Contents

1 Introduction

Let's start by setting some expectations straight. I am but a humble programmer with a background in the field of computer vision, and a passion for (pragmatic) functional programming. I do not have any background in finance. Nothing in this document is financial advice, it's only the result of my observations and thoughts around DEXs. Finally, my experience in crypto comes from experimenting with many blockchains since early 2022, mainly Cardano, Avalanche, Fantom, Algorand.

The goal of this report is to express some of my thoughts on how we could create a more efficient DEX for Cardano. I will use a somewhat narrative style for readability but still try to stay concise. I aim to go as far as my current knowledge of Cardano enables me to go and welcome feedback on any inaccuracy or wrong information this may contain.

First I will report my experience on the multiple DEX-related projects I've interacted with and explain their workings. Then I will outline the key concepts I would like to explore for a new DEX on Cardano and also try to detail as much as I can a path towards an implementation of such a DEX on Cardano. Finally, I will try to run simulations showing how an example implementation behaves compared to other existing approaches.

2 Experience Report on Other DEXes

2.1 Discovering DeFi in 2022

My first experience with DeFi dates back to early 2022, when a friend suggested I take a look at some strange unicorn-themed white paper (Uniswap [1]) , some stranger farming-themed website (Trader Joe [2]) and an even stranger cow-themed website with unbelievable returns displayed (Beefy Finance [3]). Boy I was not prepared! I fell in all the traps one can imagine. I tried Uniswap by sending some Eth from a CEX to Metamask, only to realize network fees cost more than my whole swap. I tried bridging Eth to Avalanche only to realize I couldn't use Trader Joe without some Avax to pay for fees. I tried swapping assets with transactions failing. I tried providing liquidity only to suffer from impermanent loss (remember I started at peek bull market). I tried yield farming on tokens that would lose value faster than any farming could compensate. So many times I thought I had lost a chunk of what I was trying, only to realize it was just present in my wallet in another form that was not displayed.

I think you get the gist, DeFi user experience is a disaster. And since 2022, it hasn't improved much. Of course a big part of it is just blockchain-related, and not DeFi-specific, but if possible, let's make technical choices that improve UX, not the opposite. That's where I think chains with native assets will shine, and also why I'm interested in Cardano, in addition to FP and research-backed.

Small parenthesis, I think what Radix is doing with their transaction manifest is great for UX and other chains should take some inspiration from it.

2.2 Constant product AMMs

Automated market makers (AMM) decentralized exchanges are exchanges where the platform aggregates all the assets and automates trades by providing a price algorithmically. AMM DEXs were popularized by Uniswap [1] with the constant product AMM (details later). There exists constant product pools with 2 or more assets, and with different liquidity ratios, such as 50/50, 80/20, 25/25/25/25 [4, 5]. The standard 50/50 ratio is the most efficient, while more disproportionate ratios like 80/20 suffer more slippage but less impermanent loss. These are rather imple AMMs, so we can take a little time explaining how they work.

Let's consider a pool, composed of 50% btc, and 50% eth. Let `btc_count` and `eth_count` be the amount of btc and eth tokens respectively in the pool. The constant product rule enforces that `const = btc_count * eth_count` stays constant as long as no liquidity is added or removed to the pool. Therefore the reserves of each token follows an inverse curve, as displayed in Figure 1.

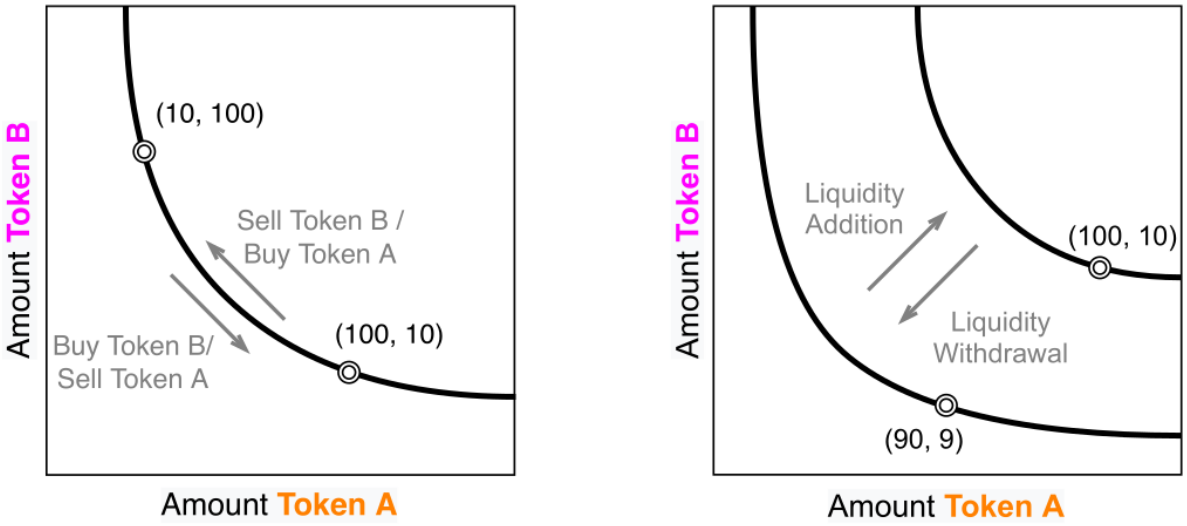


Figure 1: Liquidity curve of a constant product AMM. Figure from Xu et al. [6].

If someone wants to swap, say `btc_swap` to ETH, the following would happen, in pseudo code.

```
# Pseudo-code describing a swap from BTC to ETH
const = btc_count_before * eth_count_before
btc_count_after = btc_count_before + btc_swap
eth_count_after = const / btc_count_after
eth_swap = eth_count_before - eth_count_after
```

The `eth_swap` you get is thus obtained not at the price defined by the tokens ratio before your swap, but at the price defined by the ratio after the swap. The ratio of amount you get versus what you would have gotten at spot price is called slippage. Let `$B` be the price of BTC and `$E` the one of ETH. Considering BTC and ETH ratios are both 50% in the pool (same value), we have `$B` and `$E` linked by the equation `btc_count * $B = eth_count * $E`. Let `eth_swap_spot` be the theoretical amount received at spot price without slippage. We thus have `eth_swap_spot = btc_swap * $B / $E` and `slippage = eth_swap_spot / eth_swap`. Considering `btc_count` and `eth_count` are linked via the constant product rule, we can express the price of ETH related to the one of BTC with `$E = $B * btc_count / eth_count`. Let `swap_ratio` be `btc_swap / btc_count_before`. Then we have the following equations.

```
eth_swap_spot = swap_ratio * eth_count_before
eth_swap = eth_count_before - (const / btc_count_after)
# replacing const by eth_count_before * btc_count_before:
eth_swap = eth_count_before * (1 - btc_count_before / (btc_count_before +
btc_swap))
# Dividing eth_swap_spot and eth_swap by eth_count_before, slippage is:
slippage = swap_ratio / (1 - btc_count_before / (btc_count_before + btc_swap))
         = swap_ratio / (btc_swap / (btc_count_before + btc_swap))
         = swap_ratio * (btc_count_before + btc_swap) / btc_swap
         = (btc_count_before + btc_swap) / btc_count_before
slippage = 1 + swap_ratio
```

So the slippage of a 50/50 constant product AMM pool is proportional to the ratio we want to swap compared to the available liquidity in the pool. For example, swapping $100K in a pool containing $400K will increase the trade price by 25% while an order book with $400K liquidity might cause a slippage of only 10% or 5% if the liquidity is sufficiently concentrated around the spot price. Therefore, slippage makes constant product AMM pools less competitive, driving less volume and less fees compared to typical order books where liquidity is concentrated around the spot price.

Since 2022, I've tested the following constant product AMM DEXs.

- On Ethereum: Uniswap
- On Avalanche: Trader Joe, Pangolin Exchange
- On Fantom: Spooky Swap, Spirit Swap, Beethoven X, Solidly
- On Moonbeam: StellaSwap, BeamSwap
- On Algorand: Pact
- On Cardano: SundaeSwap, Minswap, WingRiders

In general, the user experience on all the constant product DEXs on EVM chains is roughly the same. Performing a swap is usually straightforward. Pick two tokens, enter the input value, get a quote and execute a trade. For EVM chains, I've had the best experience on Avalanche, Fantom and Moonbeam as transactions are finalized in a few seconds. The good points I've observed on these platforms are the following.

- Swaps execute immediately (within a few seconds).
- Swap prices are usually cheap as you only pay for the transaction gas (plus swap fees) making them suitable even for small value swaps.
- DEXs support complex routing. Swaps are composable, meaning if you want to swap A for C but the A/C liquidity is very low, you can instead swap A/B followed by B/C if both have more depth (B is usually USDT or the chain native token). All is done in a single transaction calling all the relevant smart contracts together.
- Providing liquidity is straightforward as many offer a "zap-in" feature where one can enter with only one side, avoiding the need to balance token ratios before.

However, they also have hurdles.

- Trading a token requires signatures authorizing the DEX to modify your balance. Gas costs encourage unlimited acces authorizations to avoid additional transactions. This is a high security risk if the DEX is exploited.
- Swaps can fail, as any transaction can fail, while still paying the gas cost.

- Gas prices tend to increase abnormally in periods of high activity, rendering small swaps unviable as they become prohibitively expensive, ranging from a few cents to tens of dollars (Figure 2).
- Most DEXs do not support limit orders. This often is because the EVM paradigm consists in putting all the contract code on chain, which makes recording an order book prohibitively expensive.
- Native tokens deposited in pools cannot be staked.
- Liquidity farming can get confusing with all the LP tokens. This point is true for all chains.
- Swaps are easy targets of sandwich attacks taking profit from slippage settings, due to transaction prioritization with gas fees.

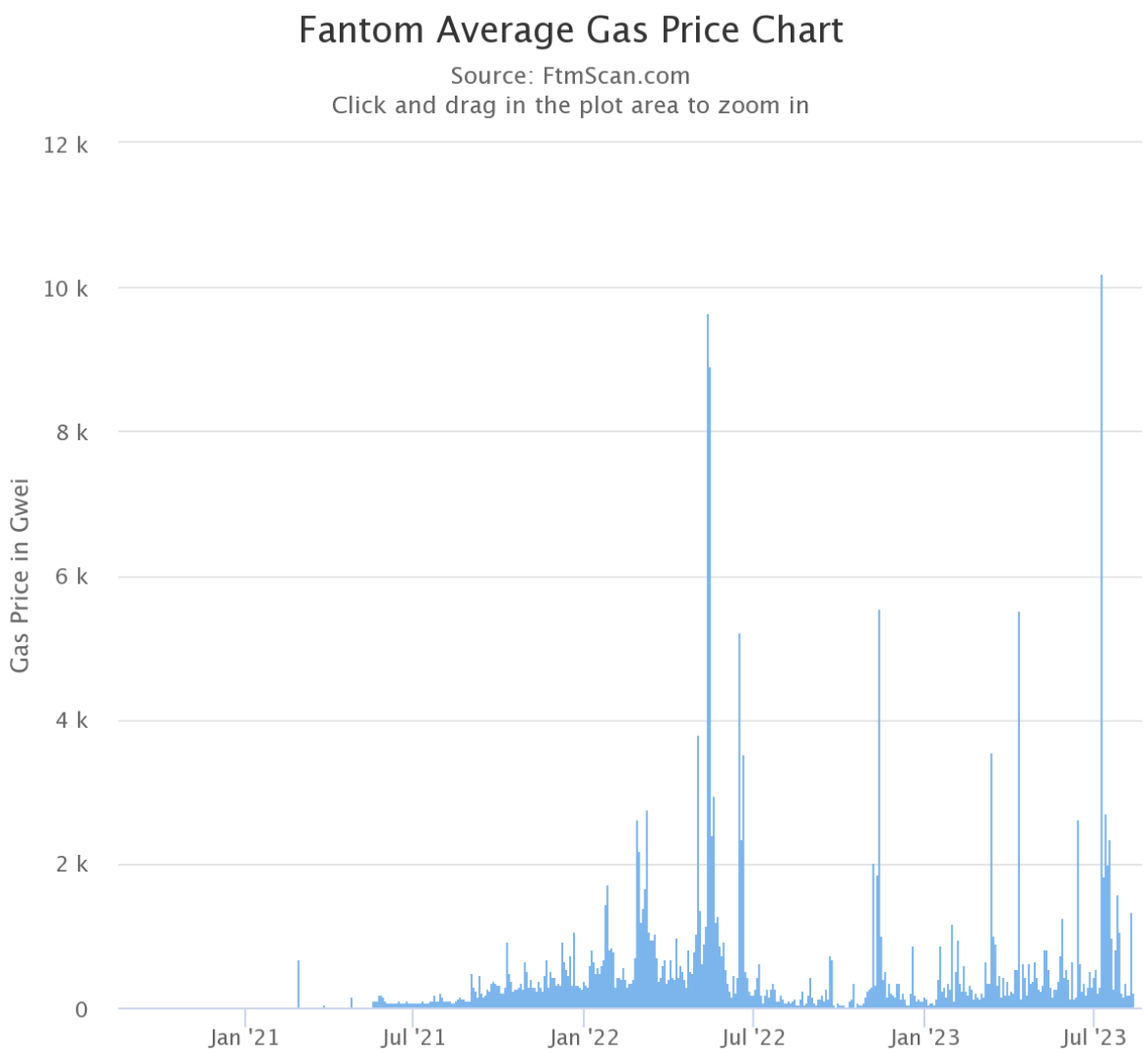


Figure 2: Fantom average gas price.

In contrast, the programming model of Cardano is very different from EVM chains. It is based on the eUTxO paradigm, and provides a virtual machine based and lambda calculus. Smart contracts are not turing complete script modifying a global state. They are predicates (pure functions returning a boolean) verifying that the set of inputs and outputs follow the rules given by the contract. Another important note is that transactions are deterministic. *Given a set of existing inputs*, we can guarantee that a valid transaction will be executed as such. I thought

important to state these differences as they are the root of some of the advantages and issues of Cardano AMM DEXs.

Here are the main good points I found while using Cardano AMM DEXs.

- All tokens are native to the platform, no need to give unlimited token access to the DEX.
- The user interface is usually well polished, providing price and volume historic data.
- Limit orders are possible.
- The Ada pooled is delegated ("staked"), both providing yield and securing the chain.

Now let's explain the root cause of most of Cardano AMM DEXs issues. One fundamental architectural constraint is that transactions are deterministic. Which in turn requires an explicit selection of UTxO inputs. Which in turn leads to contention on the UTxO holding the aggregated liquidity of a pool, consumed at each swap. To circumvent this problem, current Cardano AMM DEXs employ a 2-step solution.

1. You submit an order describing your intention. For exemple the intention to swap iBTC for Ada, with a slippage limited to 1% according to current spot price.
2. Bots, called "batchers", gather all orders and execute them.

And so, here are the main hurdles encountered on Cardano AMM DEXs.

- Swaps take very long to be executed. It took 9 days at SundaeSwap launch! Now most orders are executed within a few minutes, except in periods of high trading activity where it could take longer, due to batcher limitations.
- It's a 2-step process. The executed transaction for the order does not return the swapped counter party, you have to wait for the order to be executed by a batcher.
- Fees are complex and expensive. They include at least 4 lines, one for the chain fees, around 0.2 ada, one for the swap fees, around 0.3% of the swap amount, one for the batcher fees, around 2 to 3 ada, and one for a deposit which is returned when the order is executed or cancelled, around 2 ada.
- The batching process is usually permissioned, and may contain bad actors profiting from orders reordering.
- Delegation of the aggregated pooled Ada requires governance.

As of today, August 2023, some of these issues are now or soon going to be mitigated, but not eliminated. For example, the new Spectrum AMM DEX brings permissionless batchers. And new programming languages for Cardano, such as Aiken [7], will most likely reduce batcher fees to 1 ada or less.

Finally, the Algorand DEX Pact provides a user experience that shares some of the benefits of EVM chains and Cardano. It has very fast finality, few seconds, and the cheapest transactions. It also has native assets so we don't need to give unlimitted spending permissions. However I've found the transaction signing unwieldy. A transaction is usually composed of many small transactions composed together, so we need to sign multiple times when interacting with algorand smart contracts. The understanding of which asset goes where is also often counter intuitive. On that aspect, it's similar to Cardano UTxOs, that may be hard to follow when looking at a transaction. Generally, I found the experience on Algorand better than any EVM chain on all points.

2.3 Other Constant Function Protocols

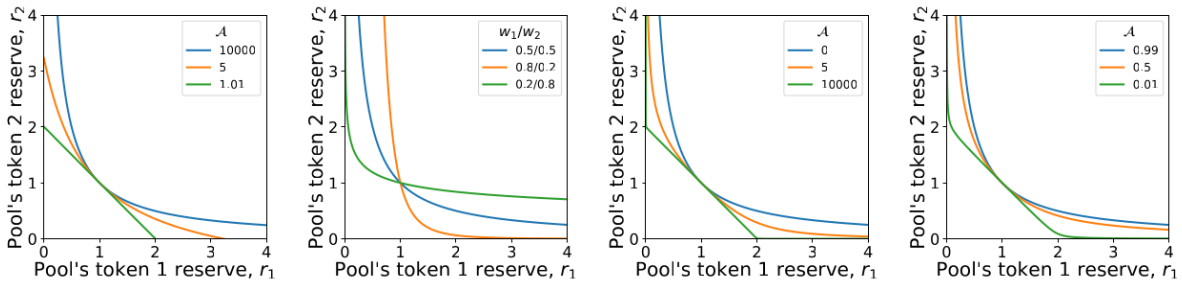
The survey by Xu et al. [6] does an excellent job summarizing the space of constant function AMM DEXs so I suggest you read that paper for a more in depth understanding. The gist of it

is that an AMM usually serves as an intermediate party, gathering all liquidity and providing a price via a function maintaining some properties. The function doesn't have to be the constant product, as with Uniswap V2. For example, using a contant sum instead, ensures that the price of two assets stays the same, which is well suited for stable pairs. However, the liquidity is then bounded by the available amount of each asset, and one side can be entirely emptied, preventing further usage of the pool. So usually, more complex functions are used. The goal is almost always improving the efficiency of a DEX, by reducing the slippage and the divergence loss (sometimes called impermanent loss).

In general, such a function has to be nonnegative and convex. The different properties to take into account are the following.

- Is it possible to design zero-impact price of liquidity changes (deposits and withdrawals)?
- Is it output-bounded? Token can be depleeted or not depending on if the liquidity curve goes to infinite and 0+ or not.
- Is it liquidity sensitive? How slippage increase with low liquidity?
- Is it demand sensitive? How price inceases when demand of token rises?

For example, Uniswap V2 is output-bounded, liquidity sensitive and demand sensitive. Uniswap V3 [8] is not output-bounded. Curve [9] is not liquidity sensitive and not demand sensitive as long as one side is not close to being emptied, but becomes both when reserves get too unbalanced.



(a) Uniswap, Eq. 27 (b) Balancer, Eq. 36 (c) Curve, Eq. 44 (d) DODO, Eq. 49

Figure 3: Conservation functions of some AMMs with different adjustable parameters to control liquidity concentration. Figure from Xu et al. [6].

2.4 Liquidity Book

We have identified that concentrated liquidity is possible with AMMs by adjusting the conservation function. One such example is Uniswap V3 which enables restricted ranges where one's liquidity can be used, instead of infinite ones. However this approach is still incompatible with limit orders that typical order books are capable to handle. Thus we are starting to observe new DEXs with functionalities much closer to order books.

On EVM chains, the main difficulty to implement order-book like DEXs, is the necessity of keeping the book of existing orders. Since the EVM paradigm encourages to put all the code on-chain, keeping track of an order book state becomes too expensive to be viable. However, if the orders are aggregated, and limited to a sufficiently small number of prices, it becomes manageable. This is the insight that some new protocols such as iZiSwap [10], Trader Joe Liquidity Book [11], and Dusa [12] are now exploring.

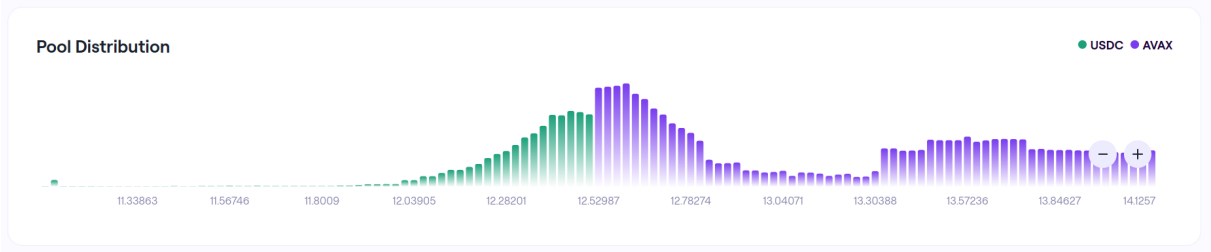


Figure 4: Example liquidity distribution in Trader Joe liquidity book.

By taking a snapshot of the liquidity distribution in Trader Joe's Avax/USDC pool, we can spot some interesting patterns. In Figure 4, each bin corresponds to a static Avax/USDC price. As defined by the pool configuration at creation, bins in that pool are separated by price steps of 0.2%. First, we observe a bell curve concentrating liquidity around the current spot price. Then we observe a clear pattern of liquidity DCAing out by taking profits at Avax prices around the $14 range. This is another advantage of the liquidity book over conservation functions, market makers can engage in more diversified strategies.

The main drawback of a liquidity book, is the necessity for market makers to frequently adjust their liquidity in order to stay relevant with the current market price. In contrast, a constant product AMM can be considered as a "set-and-forget" strategy. This is because the conservation function is an automatic rebalancing algorithm where the rebalancing price is paid by the traders. And as long as you are able to wait until the spot price comes back close to your entry point, you will not suffer from impermanent loss. The only issue is that compared to more efficient markets, constant product AMMs generate less fees, thus less APR and do not compensate for impermanent loss. This is the reason why all constant product AMMs also distribute "farming" rewards, in the shape of their governance tokens.

In the liquity book model, divergence loss does not stay "impermanent" as soon as a rebalance is operated as the loss is then realized. So to counter this issue, Trader Joe introduced a dynamic volatility fee, which increases fees in periods of high volatility.

Finally, in order to also provide a "set-and-forget" solution to liquidity providers, Trader Joe deployed a bot service called "auto pools". This is a liquidity provider strategy that you delegate your assets to and manages all the rebalancing for you. The main difficulty is finding the right strategy. Indeed, as each rebalance effectively realize a loss it needs to be well thought out. With this auto pool share price (Figure 5), we observe that it has in fact been losing value for the past 30 days.

Figure 5: Share price historic data for "the General" auto pool for Avax/USDC.

3 Security, Privacy, Scalability

Ideally, a DEX should be 100% secure, maintain privacy, and scale with users. However there are tradeoffs for each of these subjects. Let's discuss these.

3.1 Security

Security consist in ensuring that user funds are safe and cannot be misused or stolen. The security of a protocol depends on different levels. At the top level, the chain itself needs to be secure. At the protocol logic level, all possible usage of the protocol must be anticipated and evaluated. At the code level, there must not be any bug, or backdoor. Today, the most common attacks on DEXs are the following.

- Infrastructure layer attacks: destabilize the underlying blockchain, or offchain infrastructure of the exchange, for example DDOS (distributed denial-of-service) attacks.
- Time manipulation attacks: any dependency on time in contracts is an attack surface if time manipulation is possible at the infrastructure on application layers.
- Transaction reordering attacks: for example "sandwich attacks" consisting in placing new orders just before and after other orders to change prices and pocket the slippage.
- Vampire attacks: rapid economic incentives to move all liquidity out of a protocol into another one, such as what SushiSwap did on Uniswap.
- Oracle attacks: if parts of the protocol depend on reliable outside feeds, such as price oracles, it may be possible to attack these instead of the DEX directly.

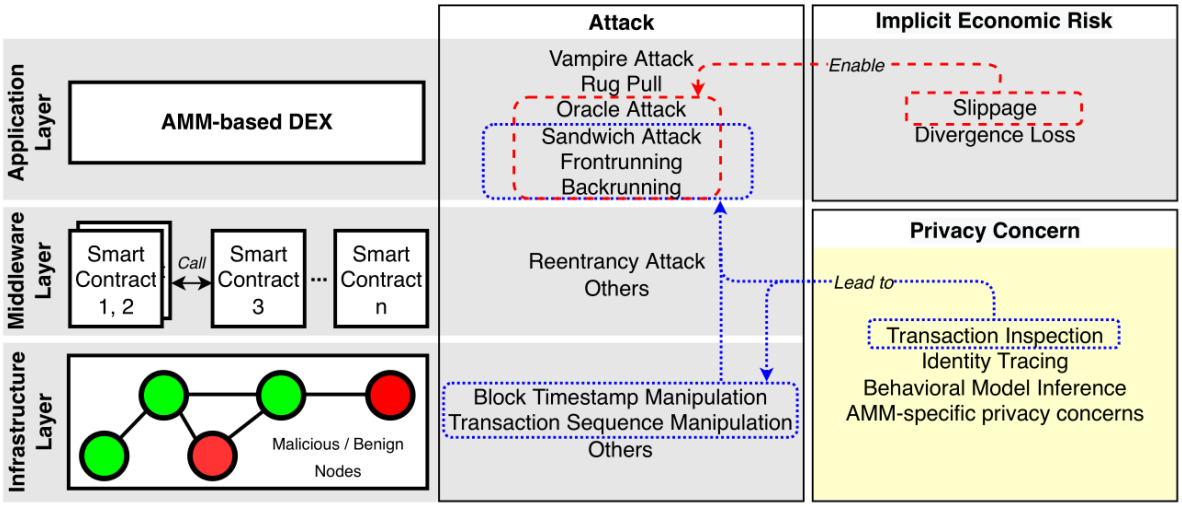


Figure 6: Common risks and attack vectors of AMM DEXs. Figure from Xu et al. [6]

3.2 Privacy

Privacy on a DEX is a complex matter as all interactions are publicly recorded on-chain. As a result it is usually possible to trace the movement of funds and link addresses together. Transaction inspection is also a factor enabling transaction reordering attacks. If nobody could inspect a swap before it is executed, no reordering attacks could happen. Current research on the subject of DEX privacy consists in using zero-knowledge layers, but this is too far from my current focus so I'll leave it for later or for someone else.

3.3 Scalability

All crypto users think about TPS (transactions-per-second) when talking scalability. However, it is an incomplete and unreliable measure. Unreliable because not all transactions are equal. Some may contain a single value transfer, while others can group many. In the context of a DEX though, it would make sense to measure the throughput in terms of swaps per second or liquidity operations (deposit, withdrawal) per second.

Focusing only on throughput though gives an incomplete picture. As the user base grows, a DEX should scale accross all the following aspects.

- Throughput: how many transactions the DEX can complete per unit of time
- Latency: how long until a transaction is executed
- Fees: how expensive are the transactions
- Reliability: how often does a transaction fail, and at which cost

4 Other Financial Activities

In traditional finance, there exists a large panel of trading products. In contrast, DEXs are usually very limited in scope. Sometimes they don't even offer limit orders. I'll try to outline these features, limited to what I think I understood. It will be interesting later to design a DEX with some of these capabilities.

4.1 The Different Types of Orders

Limit orders come in different shapes. Usually they are valid until cancelled, but it is also possible to restrict their validity to a time range. In addition, different variants of orders exist for convenience.

- Market order: immediately swap a given amount of one token for another at the current best price that the market can offer.
- Limit order: place a buy or a sell offer at a specific price.
- Stop loss sell order: trigger a sell market order if the spot price goes below a certain threshold. This protects against complete crashes by selling before it's too late. This requires the ability to restrict orders depending on context (oracle?).

4.2 Trading and Liquidity Strategies

Trading platforms often offer the ability to add trading bots. These are automated strategies placing orders for expected profits. The most famous bots are grid trading strategies. They consist in placing buy and sell orders below and above the current price in a regular (arithmetic or geometric) grid. Once a step is reached in one direction, the order is automatically switched. A regular spot grid is configured to operate within bounds and cannot operate outside as one side of its liquidities has been depleeted. In context of futures or margin trading, it is theoretically possible to have grids able to extend past their initial bounds. Figure 7 shows an example grid configuration on the KuCoin platform.

Figure 7: Grid setting on the KuCoin platform, configured to operate with 10 steps between \$0.23 and \$0.44.

In addition to trading strategies, such as grids, an order-book type DEX also enables liquidity strategies. For example, Trader Joe suggests three types of liquidity strategies when making a deposit (Figure 8). The Spot liquidity has a flat profile, the Curve one has the bell shape, and the Bid-Ask one is shaped like a V. By adjusting their range, in addition to the shape, one can easily start liquidity strategies adapted to their situation.

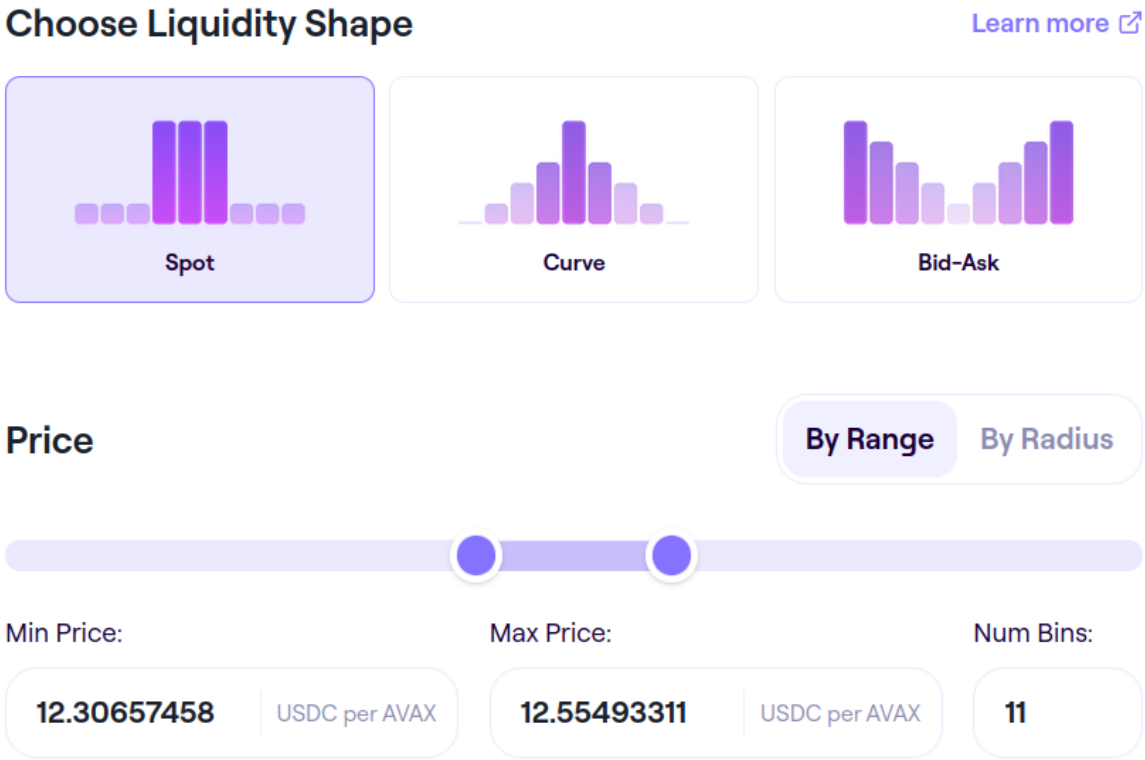


Figure 8: Suggested liquidity strategies when depositing in Trader Joe liquidity book.

4.3 Forward Contracts, Futures, Perpetuals, Options

After reviewing order types, trading and liquidity strategies, let's take a look at some more advanced trading products, based on speculation over future prices. Forward contracts and related products are derivative instruments as they are priced not directly, but indirectly compared to their underlying assets. There are two main use cases for derivatives, speculation and hedging. Let's describe some of these contracts.

- Forward contracts, and Futures, consist in agreeing in advance for a future payment. At maturity, the contract owner is forced to buy/sell at the agreed price.

- Perpetual futures are similar except they don't have a expiration date. Instead, an interest is payed regularly depending on price difference and leverage. Trading with perpetual futures can also be called margin trading, and enables leveraged trading. The Perpetual Protocol [13] and GMX [14] are example implementations on a distributed ledger.
- Options are prepaid rights, called "premiums", to buy or sell an asset at a given "strike" price, for a given duration. Contrary to futures, you have the right, but not the obligation to buy/sell. There are different kind of options depending on what you expect to happen.
  - bullish: the call, bull call spread, and bull put spread
  - bearish: the put, bear put spread, and bear call spread
  - expect high volatility: the straddle, and the strangle
  - expect low volatility: the long butterfly, and long condor

  Dopex [15] and Hegic [16] are two example implementations of on-chain options exchanges.

4.4 Borrowing, Lending, and Bonds

There are different kinds of borrowing and lending approaches, using variable or fix interest, infinite of finite duration. Since KYC isn't common on blockchains yet, lending protocols require a collateral, valued higher than the borrowed amount. When the collateral value drops below the borrowed amount, with some margin, the user is "liquidated", meaning its collateral is sold to cover the costs.

In crypto there is also a new concept called a flash loan, which is not collateralized. Instead, flash loans are given and repaid, with interest, atomically within the same transaction. They are a new DeFi primitive often used for arbitrage, liquidations and collateral swaps. Flash loans are also notorious for being used in many DeFi exploits as they allow exploiters to not risk their own funds. The most famous lending protocol in DeFi is currently Aave [17] with $5B in TVL according to DefiLlama.

TODO: give a look at TimeSwap protocol, which seems to be an oracle-less, option-like kind of protocol.

A bond is a loan emmited by a borrower looking for capital, with the promess of returning the full amount later, "at maturity", plus additional interest payed regularly. Bonds are often emmited by states or companies to crowdfund capital too big for a single party. There hasn't been many decentralized bond protocols. A few protocols in the EVM chains such as Olympus Pro from Olympus Dao failed, mostly due to ponzinomics. In Cardano though, there is a very interesting protocol called Optim Finance [18], leveraging bonds for staking power. Since assets on Cardano have both a spending key and a staking key, it is possible to safely lock tokens in a contract until maturity while transfering the staking power to the bond emitter. The bond emitter can then delegate that staking power how they wish as long as they fulfill the regular coupon payments.

4.5 Synthetics

Synthetic assets are tokenized derivatives following the price of another asset. They enable exposure to a given asset without having to hold the actual asset itself. In the cryptocurrency world, synthetics are generally used to track the price of assets existing outside of their context, such as in the real world or on other chains. Like many derivative products, they require reliable pricing sources, called oracles [19]. This is because synthetic assets are minted while locking a collateral asset of higher value. If the collateral value drops, the minter gets liquidated.

In the EVM world, Synthetix [20] is the most used platform with roughly $500M according to DefiLlama, and its oracle are provided by Chainlink [21]. UMA is an alternative, based on optimistic oracles [22]. On Cardano, we have the Indigo protocol [23], currently using custom oracles for iUSD, iBTC and iETH synthetic assets.

5 Building a new Concentrated Liquidity DEX on Cardano

You made it to here! Now the fun part begins, how do we apply all the knowledge from our study to devise a new DEX for the Cardano ecosystem. That's what I'll try to answer, by presenting a plan for a new DEX with the following properties:

- **Open source**: The contracts will be open sourced at testnet launch or before. Parts of the off-chain code will also be open sourced, especially the parts at the frontier with on-chain code.
- **Fast transactions**: An order should happen in the next block, within 20s, as long as there is space and your wallet submit it in time.
- **Direct market orders**: No need for a 2-step process where you first send tokens then get back the swapped ones if a batcher processed your transaction. What you sign is what you get (wysiwyg)!
- **Partial limit orders**: Limit orders are possible and can be partially filled.
- **Atomic composable transactions**: Routing between multiple actions, such as swap iBTC/Ada then Ada/iEth happen in a single transaction. This increases the DEX efficiency.
- **Scaling**: The DEX parallel throughput scales as more participants provide liquidity.
- **No sandwich attack**: There is no slippage. And nobody can profit from a transaction you are involved in since exact swap prices are fixed when you sign a transaction.
- **Fully decentralized**: You could submit the transactions yourself.
- **Simple low fees**: You only pay for the trading fees and a small convenience fee for the matchmaker suggesting the trade. No more deposit, and the matchmaker fees should be lower than current batcher fees.
- **Lower fees limit orders**: Limit orders are better for liquidities than market orders so they can have lower fees.
- **Automated trading**: Grid trading natively supported.
- **Automated liquidity providing**: Zap-in set-and-forget liquidity providing where rebalancing is managed by offline algorithms.
- **Fine-grained liquidity providing**: You have flexible liquidity provision, more or less concentrated, for advanced strategies.
- **You keep your staking power**: Every Ada you put in limit orders, trading bots, or liquidity providing keeps your staking key.
- **Futures, Options**: American and European futures and options are possible.

And eventually, after adding support for some type of oracle, optimistic or not, it would also be possible to add the following features.

- **Perpetual futures**
- **Borrowing** with your liquidity position or grid strategy as collateral

Quite a program, right! These are not empty promises, so let's detail what type of architecture we envision to support these claims.

TODO: check if possible to have passive sliding orders/liquidity. Seems complicated due to time range handling in a transaction.

5.1 Discrete Liquidity Board

The liquidity board is a virtual representation of all the liquidity deposited in the protocol by liquidity providers. Just like in the liquidity book approach (Section 2.4), it is a **discrete partition of the price space**, with regular geometric intervals, such as 0.1% price step increases for example. Liquidity deposited in a price bin can only be traded at exactly that price. Unlike the liquidity book approach, the **liquidity provided by each participant in a price bin is not aggreagated in a single pool**. There does not exist an on-chain data structure organizing all liquidity. This design embraces Cardano's idea of offloading off-chain anything that does not need to be on-chain. Each participant's UTxO is independant and you need offchain servers to keep track of the ledger and operate as a matchmaker. Each UTxO providing liquidity contains in its own data payload the sufficient information for validators to identify the price bin it belongs to. As a result, **anyone presenting a valid UTxO (not consumed yet) to the swap validator (smart contract) of the DEX is able to execute the swap**. This is **fully decentralized**. Of course, for convenience, most will call matchmaker APIs instead to identify the best UTxOs available for a swap. In sum, the discrete liquidity board is a peer-to-peer protocol standardized to enable offchain automated matchmaking.

TODO: add liquidity board representation

TODO: add swap diagram

5.2 Ada Staking

One very interesting property of not aggregating all liquidity of a bin is that **each UTxO can keep the delegation credentials of its original owner**. As a result all the Ada provided in liquidity still counts in your staking amount, but the amount will fluctuate as liquidity is trading back and forth. This avoids the need for governance to decide how to stake the Ada liquidity and helps secure Cardano.

5.3 No Sandwich Attack

Another interesting property of Cardano's execution model is that transactions are deterministic and identify the exact UTxOs to be consumed and created. When submitting a swap, **you sign a transaction specifying exactly at which price** and using which liquidity UTxO. Therefore, you cannot pay any more or receive any less than what you signed. The only possible attack is someone targeting that same liquidity UTxO and having the power to propagate it faster or move their transaction before in the cardano node about to produce a block. However, even in this situation, your swap transaction will fail at the first validation step, for trying to use an already consumed UTxO. Failing at this stage does not cost you anything, not even the transaction fee. So you can simply resubmit the transaction later.

5.4 Low fees

For an actor performing their own matchmaking, there are only two fee lines, (1) the Cardano transaction fee which is unavoidable, and (2) the swap fee to compensate the liquidity provider, typically 0.25% of the swap amount. When using a frontend, there will typically be an additional matchmaking fee, to compensate the matchmaker for the work of keeping track of the ledger state and picking the right UTxOs for the swap. Exact prices are unknown yet, but I suspect they will be much lower than what's currently seen with batcher fees. It could even turn into a matchmaker free market, rewarding the cheapest and most efficient ones, either directly on the official DEX frontend, or on any other frontend. Contrary to all Cardano DEXs running today, swaps are done directly. When submitting a swap you see both your token inputs and outputs,

what you sign is what you get (**wysiwyg**). There isn't a two-step process like with batchers. As a result, there is no need for a deposit to be refunded later.

5.5 Automated Liquidity Providing

The liquidity book/board model has better capital efficiency than the constant product model. However this means liquidity providers must actively manage their liquidity to take advantage of this model. Many prefer the set-and-forget fashion of constant product AMMs. For this reason, we will also release automated liquidity providing strategies. These will take the form of liquidity bots that you delegate your liquidity to and manage it for you. This will not be risk-free. As we have seen with Trader Joe's "The General" auto pool (Figure 4), rebalancing liquidity positions increases divergence loss, which might turn out more expensive than the accumulated fee. The worst case scenario is volatile market in a single direction. One way to counter this scenario could be using straddle options, to protect against rapid movement. I think we will figure out conservative strategies, and find the right optimizations to make automated liquidity providing worth it. It might take some time to validate strategies before making them available, and of course, we will need transparent advertising and performance reporting.

5.6 Trading Strategies like Grids

In essence, grid trading simply consists in managing limit orders. A grid trading strategy is initialized with the following parameters.

- Grid range: the low and high bounds for limit orders
- Total amount: the initial amount of liquidity deployed to the strategy
- Steps count or steps price increase: you can either specify the number of steps within the bounds, or the increase between two steps, like 1% for example.

TODO: add diagram explaining the grid parameters

Concretely, a regular limit order and one belonging to a grid strategy are the same for someone matching it in a market order. The only difference, is that after the swap, regular order funds are returned to the liquidity provider, while grid order funds are reused for a new limit order at a different price in the opposite direction. This can be encoded in the UTxO data as something like `Limit {priceBin = P} | Grid {priceBin = P, nextBin = P+N}`. The swap validator then simply checks that the swapped liquidity goes to the right location.

This general scheme of encoding possible swap behaviors right in the UTxO data is quite powerful and may enable more advanced strategies. Grids are the most common so it makes sense to start with them for now.

5.7 Scaling of the DEX

Scaling throughput of the discrete liquidity board depends on three things,
1. available UTxO count and value at each price bin for parallel processing,
2. off-chain computing power for matchmaking,
3. and transaction chaining to avoid waiting for the next block.

Due to market forces, an increase in trading volume will also bring new participants to liquidity proving, so (1) scales naturally. The existence of central limit order books (CLOB) on centralized exchanges, and the general availability of computing power, makes me think that off-chain matchmaking (2) will also not be an issue. Finally, the proof by the just released Spectrum DEX that batchers are able to perform transaction chaining, makes me think that (3) will not

be an issue either. But as we said previously, scaling isn't just a matter of throughput. We also need to take into account latency, fees, and reliability.

As long as UTxOs are processed in parallel (1) and transactions are chained (2), latency will increase when block limits are reached. If a block is filled, your order needs to wait another 20s for the next one. So latency should only be problematic when the whole chain is working at 100% utilization. This will remain an issue as long as the swapping logic remains on Cardano L1.

In the presented liquidity board model, prices and swap fees are predetermined in the UTxO data and validator rules, so these parts of the fees should not increase. Someone submitting their own swaps therefore can always enjoy low fees. This is different when using a frontend. Frontend matchmaking fees will be set by the matchmaker themselves so we cannot guarantee that they will stay low. What we can do is provide our own matchmaker publicly and guarantee that it will stay with low fees.

Finally let's evaluate scaling reliability. We identified previously that the main reliability attack vector consists in systematically submitting orders spending the same UTxO as another one, and bribing Cardano nodes to reorder mempool transactions. I think this vector of attack cannot be prevented, and the worst case scenario would result in roughly 50% of failed transactions. This estimate comes from the assumption that with more than 50% of stake in corruptible nodes, the Cardano network itself will be worthless. Another possible reliability issue happens when a swap tries reusing a spent UTxO by mistake. Such mistakes may happen if the UTxO is spent while preparing the transaction and waiting for signature and submission. It should be possible to evaluate that failure probability depending on few parameters such as the duration to submit a transaction, and the average UTxO consumption rate. This problem mainly manifests for human submitted transactions as in the case of machines, the submission latency should be very low. I can imagine few solutions to this problem.

- Submit limit orders instead of market orders.
- In the non-adversarial context, matchmakers preparing market orders to be submitted could publish them on a public API. Other matchmakers can then listen to that feed and avoid consuming these UTxOs.
- In the adversarial context, matchmaking could fallback to a 2-step batching as we know today. This enables low-latency machine-submitted market orders.

For user experience, we could monitor the rate of unanounced orders and display on the frontend the current amount of adversarial behavior and probability of failure of a market order.

5.8 Atomic Composition

In mathematics and programming, composition means applying one function on the results of another one. And in programming, atomic transactions are series of operations where either all of them occur, or none. So by combining these two notions, atomic composition describes the idea that we can chain smart contract executions, and either all of them execute or none. I'll try to describe different ways of achieving this.

5.8.1 The Atomic Composition Problem

Let's take the swap validator as an example when swapping Ada for iUSD.

- There are two input UTxOs, the Ada in your wallet and the iUSD in the contract liquidity.
- There are two output UTxOs, the iUSD in your wallet and the Ada in the contract liquidity.
- There are constraints such as the Ada/iUSD exchange rate, and the type of liquidity being used (limit order, grid order, liquidity provision, …).

Now let's consider that we want to chain this swap, with another swap between iUSD and Hosky. This add the following constraints.

- There are two input UTxOs, the iUSD in your wallet and the Hosky in the contract liquidity.
- There are two output UTxOs, the Hosky in your wallet and the iUSD in the contract liquidity.
- There are constraints such as the iUSD/Hosky exchange rate, and the type of liquidity being used (limit order, grid order, liquidity provision, …).

Now in Cardano you cannot chain transactions and guaranty that two successive transactions happen both atomically. For a smart contract transaction to execute successfully, all validators called in the transaction must return `True`, as explained in Aiken eUTxO crash course [24]. In addition, all validators of the transaction are executed simultaneously. There doesn't exist a notion of "intermediate state" in a transaction. If it did, we could use the iUSD output of the first Ada/iUSD swap as input for the second iUSD/Hosky swap. But in practice what happens is executing both validators in the same transaction requires the presence of four input UTxO and four output UTxO.

- There are four input UTxOs: the Ada in your wallet and the iUSD in the contract liquidity for the Ada/iUSD swap, the iUSD in your wallet and the Hosky in the contract liquidity for the iUSD/Hosky swap.
- There are four output UTxOs: the iUSD in your wallet and the Ada in the contract liquidity for the Ada/iUSD swap, the Hosky in your wallet and the iUSD in the contract liquidity for the iUSD/Hosky swap.

This is everything from but efficient. Sure we can execute both contracts in the same transaction, so paying only one transaction fees, but we lost the composition part. But don't lose hope, there are solutions to this problem!

5.8.2 Loose Contracts for Composition

So what if we only tracked two UTxOs instead of four in a swap validator? After all, when swapping Ada/iUSD, we don't really care where the Ada comes from and where the iUSD is going to! The main property we care about is that the iUSD in the bin is replaced by sufficent Ada to cover the swap price and fees. That's it, we just don't care where the Ada comes from and where the iUSD goes in the swap validator! That's the trick.

TODO: add diagram of the swap that does not care where the utxo comes from.

Well hold on! What if someone submit a transaction with no input Ada and retrieves all the iUSD output in their wallet? Yes this is valid according to our validator. But you know who is not going to be happy, the ledger! Indeed the first step of transaction execution checks that the sum of values in the consumed UTxOs equals the one of the created UTxOs. And if that person does not provide the Ada, the transaction cannot create new Ada out of fin air to put it in the swap contract liquidity.

Right, right. But what if instead of their own Ada, the swapper tries to spend a UTxO from his enemy's wallet? Well this time too the contract will fail as they do not hold the spending rights for their enemy's UTxOs.

Fine, but WHAT IF the iUSD output is sent to their enemy's? Well you got me there, this would be valid from what I currently understand. HOWEVER, this is clearly reflected in the transaction you sign. It will show that you are sending Ada and that no iUSD is coming back. So at that stage, I will just say again WYSIWYG, what you sign is what you get.

Ok so now that we have cleansed our validator of any superfluous condition, let's see how atomic composition plays out. The execution of both validators in the same contracts gives the following constraints.

- There are two input UTxOs: the iUSD in the contract liquidity for the Ada/iUSD swap, and the Hosky in the contract liquidity for the iUSD/Hosky swap.
- There are two output UTxOs: the Ada in the contract liquidity for the Ada/iUSD swap, and the iUSD in the contract liquidity for the iUSD/Hosky swap.
- There are constraints (exchange rate, …) for the Ada/iUSD and iUSD/Hosky swaps.
- The sum of all input assets equals the output ones.

So if we provide input Ada that goes into the Ada/iUSD liquidity, route the iUSD from the Ada/iUSD liquidity to the iUSD/Hosky liquidity, and retrieve the output Hosky into our wallet, everything is satisfied! We effectively performed an atomic composition of two swaps.

This method of performing atomic composition is probably the most efficient that could exist. It even enables fund-free arbitrage. Indeed, while EVM chains require to provide the funds for the first sub-transaction, atomic composition in a Cardano smart contract executes all the validators simultaneously. No need for flash loans for arbitrage supported by atomic composition. You can in theory just use the funds of any of the swaps in any other swap, and pocket the arbitrage directly. As a result, our liquidity board DEX might be the most efficient DEX possible on the Cardano ledger as any inneficiency can be arbitraged cheaply.

TODO: check a few more things, such as

- what happens if as input to swap I provide another UTxO owned by the smart contract? Is the ledger ok with this? Does it even make sense? Oh man I need to learn Aiken for real ^^
- are there other attack risks of providing something they don't own?

5.8.3 The Two-Steps Atomic Composition Ghost Token

TODO

5.9 Futures and Options

Both futures and options are agreements ahead of time for the buy or sell of an asset at a predetermined price. For futures, the contract owner must execute the buy/sell at expiration, while for options the contract can, but may not, execute the buy/sell at expiration. For this flexibility of not being forced to execute the option at expiration, the option issuer pays a "premium". Finally, we distinguish between American and European style options. European style options can be exercised only at expiration, while American style options can be exercised at any time before expiration.

European style option	**American** style option
may exercise **at expiration**	may exercise **anytime before expiration**

5.9.1 Futures in Crypto

In the context of crypto, there are only two ways to guarantee a future payment will happen, either by locking the funds in advance, or by keeping track of a collateral value that must stay higher than said future payment. Without the ability to track collateral value, future contracts only make sense if the locked funds are lower than the current spot price. However there are always two parties in such contract, so one party would agree to lock funds for a worse rate than the current spot price. For this reason, futures do not make a lot of sense in the context

of trustless crypto. Once a collateral tracking mechanism is in place though, perpetual futures, which are futures without expiration dates, start getting useful as a leverage tool.

5.9.2 Options in Crypto

With options however, the option issuer isn't required to execute, and therefore does not need to lock funds. In return for this perk, the issuer pays a premium directly at issuance.

TODO: do we care about european options? could we enable any start and end date? TODO: how to make options liquid? investopedia options: https://www.investopedia.com/options-basics-tutorial-4583012

5.10 Oracle-Free Option-like Money Market

TODO: See if I can understand and adapt something like TimeSwap.

5.11 Oracle-enabled features

Using a combination of

- optimistic oracle for every claim, associated with bounty higher than the potential gain
- low frequency "true" oracle to settle

5.12 OSS and Revenue Generation

Explain how viable is the project, what are the different revenue sources.

6 Experiments on Simulations

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

7 Acknowledgments

@ktorz

Bibliography

[1] H. Adams, N. Zinsmeister, and D. Robinson, "Uniswap v2 Core," 2020. Accessed: Aug. 2, 2023. [Online]. Available: https://uniswap.org/whitepaper.pdf

[2] "Trader Joe: Decentralized Exchange." https://traderjoexyz.com/ (accessed: Aug. 2, 2023).

[3] "Beefy Finance: Yield Optimizer." https://beefy.finance/ (accessed: Aug. 2, 2023).

[4] F. Martinelli, and N. Mushegian, "A non-custodial portfolio manager, liquidity provider, and price sensor," 2019. Accessed: Aug. 2, 2023. [Online]. Available: https://balancer.fi/whitepaper.pdf

[5] "80/20 Balancer Pools." https://medium.com/balancer-protocol/80-20-balancer-pools-ad7fed816c8d (accessed: Aug. 2, 2023).

[6] J. Xu, K. Paruch, S. Cousaert, and Y. Feng, "Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols," *ACM Comput. Surveys*, vol. 55, no. 11, pp. 1–50, Mar. 14, 2023.

[7] "Aiken: A modern smart contract platform for cardano." https://aiken-lang.org/ (accessed: Aug. 10, 2023).

[8] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, and D. Robinson, "Uniswap v3 Core," 2021.

[9] M. Egorov, "Stableswap-efficient mechanism for stablecoin liquidity," 2019. Accessed: Aug. 10, 2023. [Online]. Available: https://wikibitimg.fx994.com/attach/2020/10/189869321/WBE189869321_21425.pdf

[10] J. Yin, and M. Ren, "Iziswap: Building decentralized exchange with discretized concentrated liquidity and limit order," 2021. Accessed: Aug. 10, 2023. [Online]. Available: https://github.com/izumiFinance/iZiSwap-core/blob/72589860773ce7f0b748c15b305607af450d5164/whitepaper/iZiSwap%3A%20Building_Decentralized_Exc

[11] MountainFarmer, Louis, et al., "JOE v2.1 liquidity book," 2022. Accessed: Aug. 10, 2023. [Online]. Available: https://github.com/traderjoe-xyz/LB-Whitepaper/blob/9d259d99f904b231850e113d
Joe%20v2%20Liquidity%20Book%20Whitepaper.pdf

[12] L. JOUANNY, P.-L. HABOURY, et al., "Dusa protocol whitepaper," 2023.

[13] "Perpetual protocol." https://perp.com/ (accessed: Aug. 13, 2023).

[14] "GMX | decentralized perpetual exchange." https://gmx.io/ (accessed: Aug. 17, 2023).

[15] "Dopex: Decentralized options exchange." https://docs.dopex.io/ (accessed: Aug. 13, 2023).

[16] M. Wintermute, "Hegic: On-chain options trading protocol on ethereum powered by hedge contracts and liquidity pools," 2020. Accessed: Aug. 13, 2023. [Online]. Available: https://github.com/hegic/whitepaper/blob/942acb6aa72ae3eda913b52795c3c0bcc6a8cc71/Hegic%20Protocol%20Whitepaper.pdf

[17] Aave Team, "Aave: Protocol whitepaper v1.0," 2020. Accessed: Aug. 13, 2023. [Online]. Available: https://github.com/aave/aave-protocol/blob/efaeed363da70c64b5272bd4b8f468063ca5c361/docs/Aave_Protocol_Whitepaper_v1_0.pdf

[18] "Liquidity bonds - optim finance." https://optim-finance.gitbook.io/optim-finance/products/liquidity-bonds (accessed: Aug. 13, 2023).

[19] G. Caldarelli, and J. Ellul, "The blockchain oracle problem in decentralized finance — a multivocal approach," *Appl. Sciences*, vol. 11, no. 16, Aug. 18, 2021.

[20] "Synthetics litepaper." https://github.com/Synthetixio/Synthetix-Gitbook-Docs/blob/4a5783ff71ae37cb5dde0779ea2be46c2ef8e034/synthetix-protocol/the-synthetix-protocol/synthetix-litepaper.md (accessed: Aug. 14, 2023).

[21] L. Breidenbach, C. Cachin, et al., "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," 2021.

[22] H. Lambur, A. Lu, and R. Cai, "Uma data verification mechanism: Adding economic guarantees to blockchain oracles," 2020. Accessed: Aug. 14, 2023. [Online]. Available: https://github.com/UMAprotocol/whitepaper/blob/9ae11f3bd2aef3ed37f39a05455eef85221c776a/UMA-DVM-oracle-whitepaper.pdf

[23] Indigo Laboratories, "Indigo: Synthetic assets on cardano," 2022. Accessed: Aug. 14, 2023. [Online]. Available: https://indigoprotocol.io/paper

[24] "Aiken | Eutxo crash course." https://aiken-lang.org/fundamentals/eutxo (accessed: Aug. 15, 2023).