

Arcadia Finance [Draft]

November 2023

Thomas Smets
Arcadia Finance
Brussels, Belgium
thomas@arcadia.finance

ABSTRACT

Arcadia Finance introduces a novel on-chain architecture, the Arcadia Finance Technology Stack, designed for efficient collateral management of margined positions.

Our overarching vision is to build standardized on-chain infrastructure for collateral management, unlocking the capital in diverse DeFi assets, and establishing a comprehensive platform for managing collateralized positions.

As we anticipate the continued growth of tokenized assets and emergence of new on-chain financial primitives, the importance of standardized and efficient collateral management becomes increasingly paramount.

Arcadia strives to enable higher capital efficiency for users, transparent risk assessment, faster go-to-market time for new protocols and an improved user experience. All without compromising on security, self-custody and decentralisation.

KEYWORDS

Collateral Management, DeFi, Margin, EVM

1 INTRODUCTION

This paper will explain the rationale why we are building the Arcadia Finance Technology Stack, what its main benefits are, and how it is technically implemented.

The Arcadia Financial Technology Stack is a set of on-chain, interlinked financial protocols, centered around collateral management and margined positions. They facilitate non-trusting Debtors and Creditors to close financial contracts without the need of trusted intermediaries.

It consists out of three protocols, each with distinct responsibilities and permissions:

- The Arcadia Registry.
- The Arcadia Accounts.
- The Generalised Creditor.

To establish a shared understanding, we start this paper with Section 2, where we define concepts and terminology related to collateral management and margined positions.

Section 3 provides a brief exploration of the historical use of collateral in the financial sector, with a focus on blockchain-based solutions.

Following this, Section 4 goes into more detail how collateral is used within DeFi, pinpointing current shortcomings and inefficiencies, and discusses how novel architectures might address these issues.

In Section 5 we introduce our implementation of such a novel architecture: the Arcadia Financial Technology stack.

Finally, each of the underlying protocols of the Arcadia Financial Technology stack are discussed in detail in Sections 6, 7 and 8.

2 TERMINOLOGY

COLLATERAL: The assets pledged by a Debtor to a Creditor to cover the credit risk in case the Debtor would default.

DEBTORS AND CREDITORS: We will often use the terms Debtors and Creditors throughout this paper, our definition is broader than the commonly accepted context within traditional debt arrangements. We use the term Debtor for any holder of a financial instrument that results in liabilities on the balance sheet (borrowers, option writers, payers of cash-settled futures contract or swaps...). While the Creditor refers to the entity to whom the liability is owed.

MARGIN: The value of collateral assets that a Debtor must hold in a Margin Account to cover the credit risk of its Creditor(s). The margin requirements, are set by the Creditor. It is the value of assets that is "locked" when a Debtor opens a position.

MARGIN ACCOUNT: Our definition of a Margin Account extends beyond the conventional usage within Brokerage Accounts. A Margin Account is an account held by a Debtor that mitigates the counterparty risk, borne by the Creditor. It does so by guaranteeing that the total value of all assets within the Account is always bigger than the liabilities owed to the Creditor.

OPEN POSITION: The size of the liability that a specific Debtor has with a specific Creditor.

3 COLLATERAL MANAGEMENT

3.1 Origin

Evidence of collateralized loans, also known as secured loans, can be traced back to at least 400 BCE in Ancient Greece [7]. In these early transactions, borrowers provided tangible assets as collateral to secure loans, laying the foundation for the concept of collateralization.

3.2 Current State

Over the past decades we have witnessed significant advancements in collateral management practices.

With the rise of complex financial instruments and the globalization of markets, the role of collateral has expanded beyond traditional lending. Collateral is now used to secure a wide range of financial instruments, including derivatives trading, margin accounts, and other sophisticated financial contracts.

The digitalisation of the financial system greatly improved the efficiency of the exchange and settlement of collateral. Reporting, margining and reconciliation processes could be executed on a daily basis instead of on a weekly or even monthly basis [9].

The emergence of blockchain technology and decentralized finance (DeFi) led to a new wave of innovation for collateral management. Blockchain has some excellent properties with regards to collateral management:

- With its immutable and transparent ledger, collateral can be audited 24/7.
- Smart contracts enforce margin requirements real time on a 24/7 continuous basis.
- Atomic execution of transactions avoids expensive reconciliation processes and eliminates settlement risk.
- Atomic execution of transactions enables optimistic execution of transactions (e.g. flash loans), which can greatly improve processes such as refinancing liabilities or even enable completely new financial use-cases.
- The permissionless nature of blockchain and Dapps (Decentralised Applications), combined with a shared state between all those applications, make different financial applications and assets composable and inter-operable by default.

It is no coincidence that some of the first blockchain applications with product market fit, were secured lending protocols. These early pioneers, like Maker[10], Aave[11] or Compound[6], rely on collateralization as the only way to mitigate counter party risks and work with over-collateralized loans. As such they enable peer-to-peer or even peer-to-contract loans, without the need of any trusted intermediaries.

3.3 Collateral Management in crisis

Collateralization, both in the traditional financial world and in DeFi, is not without its problems. Recent crises in both industries highlight the need for better collateral management infrastructure.

Bad collateral management, rehypothecation of collateral, opaque accounting practices and outright fraud with collateralized assets, are all cited as root causes[4] behind the 2008 financial crisis. Following the crisis, a variety of major regulations were introduced (e.g. EMIR in the European Union, or Dodd-Frank in the USA). While these packages were successful in stabilizing the financial system, they have clear centralization tendencies[3]. The major beneficiaries of said regulations are the major established institutions, mandatory intermediaries for central clearing, central banks and the regulatory bodies themselves. This might paradoxically cement even greater systemic risks into the financial system.

In 2022 DeFi experienced its own financial crisis. In a few months time, many of the ecosystem's key (albeit mostly centralised) players collapsed, and removed all liquid with them. Again bad collateral management, rehypothecation of collateral, opaque protocol mechanisms, and outright fraud with collateralized assets were at the root of the problem.

A recurring trend with protocols that imploded in 2021 and 2022 is that the lack of collateralization was obfuscated behind complex protocol designs and narratives. Some of the notorious examples are:

- Synthetic stable-tokens such as Gaia-USDF, Iron-Titan and Luna-Terra.
- Olympus and its forks relying on the (3,3) model.
- Protocols where the only utility of a token is to receive more of the token itself (animal yield farms, reflective tokens...).
- Unsecured loans by market makers and CeFi players.

DeFi protocols should focus more on collateralization, and less on Complex tokenomics that obfuscate who ends up paying when things go bad.

3.4 The Future

The over-collateralized DeFi protocols however, operated remarkably well during the aforementioned crisis, even better than their traditional counterparts. This is well illustrated by some of the collapsed centralised entities, active in the space until 2022, that had both on- and off-chain liabilities. All on-chain Creditors were repaid in a timely and orderly fashion, as enforced by their smart contracts. While the lawsuits for the many off-chain Creditors are, at the time of writing this paper, still ongoing (and probably will be for many years).

It is for these types of crises that liabilities are secured in the first place. These events further strengthened our vision that blockchain-based protocols have the potential to become the dominant infrastructure for collateral management. But before we get there, a number of Inefficiencies of current DeFi protocols must be overcome, as we will discuss in Section 4.2.

4 COLLATERAL MANAGEMENT IN DEFI

Almost all DeFi protocols use collateralization to manage counterparty risks between different user-groups in some way of form. Lending protocols, perpetual protocol, Prime brokerage protocols, option protocols etc. all require some or all users to deposit collateral.

4.1 General Principles

Most DeFi protocols work with over-collateralization to mitigate counterparty risks. Meaning that the sum of all collateralized assets must always be bigger, with a certain safety margin, than the sum of all liabilities.

For example, if a user wants to borrow 800\$ worth of token A via a certain lending protocol, he/she must first deposit and lock at least 1000\$ worth of token B as collateral in said protocol.

Most protocols have some kind of liquidation mechanism in place to prevent bad debt when the market would move against a user with an open position. Using the previous example, if token B drops in value compared to token A, say from 1000\$ to 900\$, the position of the user can be (fully or partially) liquidated.

4.2 Inefficiencies

In the next paragraphs we will highlight a number of inefficiencies, which need to be overcome before DeFi can be broadly adopted as the go-to infrastructure for collateral management.

4.2.1 Hacks and exploits. The major problem in DeFi today is the number of hacks and exploits, the estimated amount of funds lost in 2023 ranges from \$400M to \$1B. There are many security related

practices the sector as a whole should improve on, but we want to highlight one problem.

As mentioned before, almost all DeFi protocols use collateralization in some way of form. While the nature of the financial contracts may be very different for each of these protocols, they share a great amount of common logic:

- Pricing of collateralized assets.
- Management (depositing, withdrawing...) of collateralized assets.
- Margin calculations.
- Management of asset-liability specific risk parameters.
- Margin calls and liquidations of risky positions.
- Settling bad debt.
- ...

Pricing of assets, management of assets and liquidation logic is complex, error prone and most mistakes easily lead to severe user losses. Today all protocols implement these redundantly, even for new versions of the same protocol.

Not only is this very costly in development and auditing time, the core logic is rewritten and re-deployed time and time again, and with each new deployment new bugs can be introduced.

Being able to build on top of battle tested code would benefit smart contract developers, protocols, users and the overall ecosystem.

4.2.2 Fragmented and isolated collateral. The average DeFi-user has collateralized assets fragmented and isolated across a multitude of protocols, and many quality assets are sitting idle. While isolated margin positions have their benefits (they isolate risks), they are not capital efficient. Having the ability to use a portfolio of assets as collateral improves the capital efficiency for a number of reasons:

- The volatility of a portfolio of assets is always equal to, or lower than the weighted volatility of the individual assets. Or put different, assets losing in value can be compensated with assets increasing in value.
- Having less positions, with lower volatility, reduces the number of rebalancing actions and liquidations.
- Depending on the correlation between collateralized assets, the Creditor might use less strict margin requirement.
- Debtors can use negatively correlated assets to hedge positions and limit liquidation risks.

Having a global shared state across applications and assets is often cited as one of the key advantages of blockchain over centralised financial infrastructures[8]. This should open up a whole new solution space for Debtors to manage collateralized positions, and might even enable a single Debtor to share its margin between non-trusting Creditors without intermediaries. Hence it is quite ironic that traditional brokers and clearing houses offer more advanced capabilities for cross- and portfolio-margined positions, compared to the state-of-the-art DeFi protocol.

Again there are multiple underlying root causes why collateral in DeFi is still fragmented and under-utilized:

- As mentioned in Section 4.2.1, there is no shared collateral management layer, each protocol with collateral has their own non-standardized implementation.

- Protocols are build around specific assets types (eg. lending protocols for simple ERC-20 or for AMM LPs, or for NFTs). Different asset types cannot be used within the same protocol to back a single position and emergence of new primitives/token standards requires migrations/new protocols.
- Blockchain is still a young technology, core non-financial infrastructure required for on-chain portfolio management (think Account Abstraction, intents or cross-chain messaging) is only recently developed or still in development.

Fragmentation of assets not only results in capital inefficiencies, it also contributes to a challenging user experience, more on that in Section 4.2.3.

4.2.3 End user complexity. End-user do not "want" to manage collateral, they want to optimize portfolio's to achieve a certain objective (e.g. increase yield, delta hedging, diversify exposure to different protocols...). Collateral management is a means to an end, it is not an activity most users enjoy doing.

End-user adoption will only increase outside of a niche bubble of tech enthusiasts if the technical complexities around collateral management are be abstracted away. Important note, abstracting away technical complexities should never come at a cost of hiding risks or relying on centralised and custodial solutions (as is to often the case).

As mentioned in the previous Section 4.2.2, the bad user experience is partly due to the fragmentation of assets and positions. Rebalancing portfolio's often require multiple transactions per asset and per position.

Today the user has to execute multiple sequential transactions per asset, instead a single transaction that rebalances the complete portfolio. Let's take as example a user that wants to earn yield from his collateralized assets and get exposure to Liquid Staking Tokens (LSTs). Since LST protocols introduce an additional layer of risk, he wants to diversify risks over 5 different LST service providers. Assuming our user has wrapped Ethereum, he would need to do 10 transactions via multiple platforms (5 approvals and 5 swaps or 5 approvals and 5 staking actions). Rebalancing said portfolio, or depositing it as collateral, would require another 5 to 10 transactions. Not only is this time consuming, it also introduces additional operational risks, to name a few:

- Herstatt risk (settlement risk) due to changing markets before each leg of the action is settled.
- Increased risk of manual mistakes (fat fingers, bad slippage settings).
- Increased risk of falling victim phishing attacks on one of the platforms..

With Account abstraction all these different action can be bundled in a single atomic on-chain transaction.

A second abstraction is to let users define what they want, and provide them with the information (the calldata) how to do it. Since most collateral management actions require interactions with multiple assets/protocols, this abstraction can only be achieved after portfolio's as a whole can be managed with a single atomic transaction.

At the time of writing this paper, the required infrastructure to enable both abstractions, is heavily debated within the broader ecosystem and multiple proposals are launched to standardize the infrastructure. Most notable are the proposals for Account Abstraction (e.g. EIP-3074 and EIP-4337) and for Intent based architectures (e.g. EIP-7521)

Both abstractions are already successfully applied (albeit in a somewhat non-standardized form) within DeFi by Decentralised Exchanges, NFT-marketplaces and their aggregators.

4.3 New Architectures

Over the course of 2023, a number of interesting solutions were proposed by different protocols and stakeholders on how to build more robust decentralised financial systems. Notable recent papers are from Uniswap V4[1], Morpho[2] and Euler[5].

All mention similar concepts as the distinction between product and protocol, modularization of the protocol, oracle agnostic implementations and explicitly separating the logic of the protocol in different layers, where layers with different complexity evolve at different speeds.

Especially the last concept, to separate the logic in different layers, resonates with the long term vision of the Arcadia Finance team. Different Logic layers of the financial stack, with different complexities should have different life-cycles and innovation time-lines. With the lowest most core component the slowest moving, while the upper user facing layers must evolve and adapt quick in response to ever changing markets.

A good practical example is uniswap V4, where Uniswap Labs implements the underlying core mathematical logic for CLAMMs (Concentrated Liquidity Automated Market Makers). Other teams can build in a permissionless manner on top of the Uniswap V4 protocol and develop feature rich and fast innovating new DEXs, without the need and risks of redundantly implementing the complex maths related to concentrated liquidity.

Also over-collateralized protocols would benefit from a similar architecture. A shared standardized and permissionless layer with battle tested logic for collateral management and margin calculations, on top of which Creditors build their application specific financial contracts. And where fast moving end-user facing features are developed independently of the core logic.

This is exactly what we are building with the Arcadia Financial Technology stack.

5 THE ARCADIA FINANCIAL TECHNOLOGY STACK

The Arcadia Financial Technology stack, illustrated in Figure 1, is our answer to resolve the inefficiencies outlined in Paragraph 4.2. It is implemented for the Ethereum Virtual Machine (EVM) and consists out of three protocols:

- The Arcadia Registry.
- The Arcadia Accounts.
- The Generalised Creditor.

The Arcadia Registry is the least opinionated sub-protocol and implements the core pricing and risk management logic. By having a shared Arcadia Registry, Creditors can reuse battle tested core

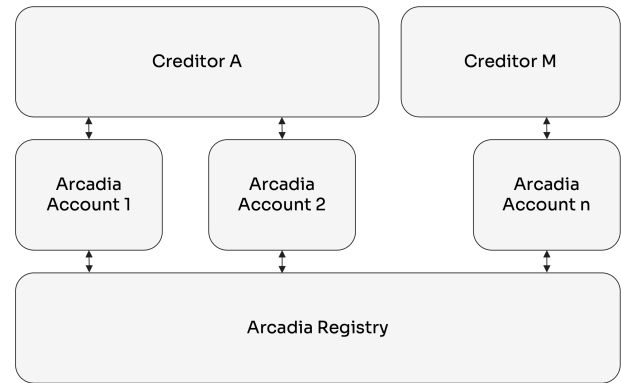


Figure 1: The Arcadia Financial Technology Stack.

logic. This not only saves development time, but avoids that bugs are introduced in redundant implementations.

The Arcadia Accounts are advanced user-controlled smart wallets, responsible for asset management and enforcing the margin requirements of their owners. With Arcadia Accounts, diversified portfolios can be used as collateral, and the same Account can serve as margin account for different Creditors.

With Arcadia Accounts, their owners can batch together multiple account actions, automate account actions and interact optimistically with third party protocols. Arcadia Accounts enable an unprecedented user experience, while not compromising on self-custody and composability.

Arcadia Accounts are further customizable to suit specific needs of certain users or Creditors.

Lastly, the Generalised Creditor can be any protocol or financial contract where one or more users have liabilities, or can have liabilities at some point. The protocol has to implement a single interface to become interoperable with the rest Arcadia Financial stack.

The protocols devs only have to implement their creditor specific smart contracts, and get the management of assets, pricing of assets, margin calculations... out of the box.

In the next Sections we will discuss each of the three different protocols in more detail.

6 ARCADIA REGISTRY

The Arcadia Registry is a non opinionated on-chain infrastructure with two main functionalities:

- (1) Standardize pricing logic to denominate a portfolio of on-chain assets in a given numeraire (unit of account).
- (2) Standardize logic to store, manage and process asset related risk parameters/variables.

6.1 Modular pricing logic

The Arcadia Registry is not a monolithic smart contract containing all logic. It consists of a main coordinating smart contract (also referred to as The Registry) and multiple append-only Modules.

Each Module is a separate smart contract with the pricing logic for specific Oracle implementations or specific Asset types. The

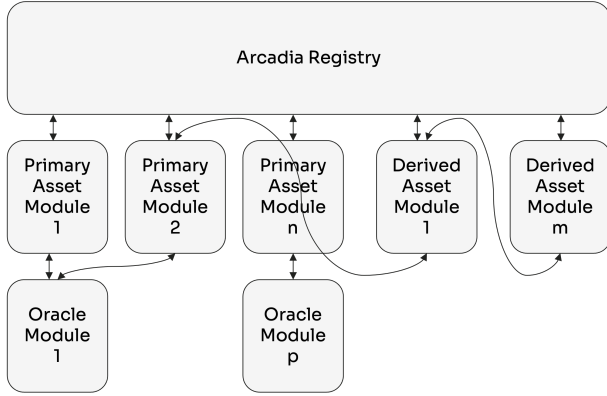


Figure 2: The Arcadia Registry and Modules.

Registry keeps mappings per Creditor of which Module to use for a certain asset.

It is the Creditor itself that sets these mappings, the Creditor decides which Modules to use for each asset they allow their Debtors to use as collateral. Different Creditors can use different Modules to price the same asset, or a Creditor can choose to not allow certain assets at all.

Modules can only be appended to the Registry, not removed or overwritten. This ensures that Pricing logic is immutable, but it still gives flexibility to add new assets, or implement more efficient Pricing logic over time.

Working with this modular architecture has a number of advantages:

- Modules are re-usable by different Creditors, instead of redundant monolithic Pricing Contracts per Creditor, battle tested Pricing/Risk logic can be reused.
- Different Modules can call each other recursively, creating Pricing/Risk logic that is as composable as tokenized assets.
- Logic for new assets, or even asset types, can be added without the need of upgradable Proxy contracts and their inherent trust assumptions.

There are two main categories of modules: Oracle Modules and Asset Modules. And Asset Modules can be further divided in two sub-categories: Primary Asset Modules and Derived Asset Modules. An overview of the different modules, and their relations are illustrated in Figure 2.

6.2 Oracle Modules

The Oracle Module implements the logic to convert oracle-rate of different oracle technologies into a standardized format. Each different oracle implementation (e.g. Chainlink oracles, Pyth oracles, Uniswap TWAPs...) has its own Oracle Module.

Every oracle has a quote-asset and a base-asset, the oracle-rate r_{oracle} reflects how much tokens of the quote-asset are required to buy 1 token of the base-asset. The precision S_{oracle} of oracles is variable and can even be a binary fixed-point number. Since all pricing logic within the Arcadia Protocol uses fixed-point numbers with 18 decimals precision, a correction from the oracle precision to a precision of 18 decimals is required for the standardized oracle

rate $r_{ba \rightarrow qa}$, returned by the Oracle Module.

$$r_{ba \rightarrow qa} = r_{oracle} \frac{10^{18}}{S_{oracle}} \quad (1)$$

6.3 Asset Modules

Just as Oracle Modules implement the logic to return oracle-rates in a standardized format, Asset Modules will return Asset values in a standardized format. And just as each oracle implementation has its own oracle Module, each asset type has its own Asset Module.

Next to pricing logic, Asset Modules also store and manage asset specific risk parameters, which we will discuss in more detail in Paragraph 6.5.

Asset Modules can be further divided into two distinct groups: Primary Asset Modules and Derived Asset Modules.

6.3.1 Primary Asset Modules. Primary assets are defined as assets that are not composed of other assets. Primary Assets must be priced using one or more on-chain oracles, for which the rates $r_{ba \rightarrow qa}$ are fetched from the Oracle Modules via the Registry.

The Primary Asset Modules will denominate all values of their assets in USD with 18 decimals precision. Since the asset amount, a_{asset} , can have a variable precision (number of decimals), a correction S_{asset} is again applied to bring the usd-value of the Primary Asset, v_{asset}^{usd} , to 18 decimals precision.

Depending of the direction of the oracle, the value of the Primary Asset is calculated different. When the asset is the base-asset of the oracle:

$$v_{asset}^{usd}(a_{asset}) = r_{asset \rightarrow usd} \frac{a_{asset}}{S_{asset}} \quad (2)$$

When the asset is the quote-asset of the oracle:

$$v_{asset}^{usd}(a_{asset}) = \frac{1}{r_{usd \rightarrow asset}} \frac{a_{asset}}{S_{asset}} \quad (3)$$

For some assets, there might not be a direct oracle to USD. For these assets multiple oracles can be chained together:

$$v_{asset}^{usd}(a_{asset}) = r_{asset \rightarrow x} r_{x \rightarrow usd} \frac{a_{asset}}{S_{asset}} \quad (4)$$

$$v_{asset}^{usd}(a_{asset}) = \frac{r_{y \rightarrow usd}}{r_{y \rightarrow asset}} \frac{a_{asset}}{S_{asset}} \quad (5)$$

6.3.2 Derived Asset Modules. Derived assets are defined as assets that are composed of one or more underlying assets (which can be primary assets or other derived assets). Examples of derived assets are liquidity positions of AMMs, staked assets, yield bearing tokens.

The derived Asset Modules will calculate the usd-value in three steps. It will first calculate the amounts of the underlying assets a_u , given the amount of the asset to be priced a_{asset} .

$$a_u = f_u(a_{asset}) \quad (6)$$

Where $f_u(a)$ is a flashloan-resistant and protocol specific function. For example for a Uniswap V2 Liquidity Positions it can be derived from the reserves of the Uniswap V2 pool contract, or for ERC-4626 tokens it can be calculated by calling `convertToAssets(shares)`.

In a second step, the Derived Asset Module will recursively fetch the usd-value of each of the underlying assets $v_{asset}^{usd}(a_u)$ from the Registry, using the underlying asset amounts a_u from Equation 6.

$$v_{asset}^{usd}(a_u) = v_{asset}^{usd}(f_u(a_{asset})) \quad (7)$$

The underlying assets of a derived asset can be derived assets themselves (e.g. staked LP tokens, or liquidity positions of AMMs where one of the tokens is a yield bearing asset). In this case, the underlying asset will in turn fetch the usd-values of their underlying asset(s) recursively. The stop condition of the recursion is when the underlying asset is a Primary asset, or if a maximum recursion depth (set by the Creditor) is reached.

By working with modular and recursive Asset Modules, the pricing logic of the Registry fully captures the benefits of composability withing DeFi.

Lastly, the total usd value of the asset, $v^{usd}(a_{asset})$, is calculated as the sum of the usd values of its underlying assets. Since all underlying assets are valued with 18 decimals precision, no additional precision correction is required.

$$v^{usd}(a_{asset}) = \sum_u v^{usd}(a_u) = \sum_u v^{usd}(f_u(a_{asset})) \quad (8)$$

6.4 Portfolio Pricing

Section 6.1 explains how the Registry can calculate the value of assets in usd. Building further on this logic, the Registry can price any combination of assets in any numeraire (the asset used as unit of account).

Given a portfolio P consisting of assets with amount a_i , the value of the portfolio denominated in a certain numeraire $v_{spot}^{num}(P)$ is given as:

$$v_{spot}^{num}(P) = \frac{\sum_i v_i^{usd}(a_i)}{r_{num \rightarrow usd}} \frac{S_{num}}{10^{18}} \quad (9)$$

A correction S_{num} is applied to bring the value from the 18 decimals precision to the actual decimal precision of the numeraire.

The result of equation 9 is the spot or MtM (mark-to-market) value of the portfolio.

Next to the MtM value of the portfolio, the Registry can also calculate risk adjusted portfolios values, where assets are discounted with a asset-Creditor specific risk factor RF_i^c , set by each Creditor. The risk adjusted portfolio value $v_{risk}^{num}(P)$ is then given as:

$$v_{risk}^{num}(P) = \frac{\sum_i RF_i^c \cdot v_i^{usd}(a_i)}{r_{num \rightarrow usd}} \frac{S_{num}}{10^{18}} \quad (10)$$

There are two important risk adjusted portfolios values: The Collateral Value v_{coll}^{num} of the Account and the Liquidation Value v_{liq}^{num} of the account.

Both are used to check the Margin Requirements of the account. The Collateral Value can be calculated by plugging the asset-creditor specific collateral factors CF_i^c of each asset into equation 10. And the Liquidation Value respectively by using the liquidation factors LF_i^c of each asset.

The Margin requirements, Collateral Value and Liquidation Value are discussed in more detail in Paragraph 7.1.

6.5 Creditor Specific Risk Parameters

Each Creditor can set a number of risk parameters in the Registry to protect themselves against different risks. These risk parameters are taken into account when depositing assets, or when risk-adjusted portfolio values are calculated.

The risk parameters can be asset specific, protocol specific or a single value per creditor.

6.5.1 Collateral, Liquidation and Risk Factors. As is described in detail in Paragraph 7.1, collateral factors and liquidation factors are used to discount asset values to account for market, liquidity and smart-contract risks.

Each asset i that can be priced by the Registry has a Creditor specific collateral factor CF_i^c and liquidation LF_i^c factor. For Primary Assets, the factors are set directly by Creditor.

For Derived Assets, the risk factors are a function of the risk factors of its underlying assets u , and optionally an additional protocol specific risk factor RF_p^c , to account for the additional complexity and smart contract risks of derived assets:

$$CF_i^c = f(\forall CF_u^c, RF_p^c) \quad (11)$$

The relation between the risk factor of the derived asset and the risk factors of the underlying assets, will depend on the specific nature of the derived asset. It can be the minimum of each of the risk factors of the underlying assets, or a weighted average or something else.

For derived assets with a single underlying asset, the risk factor becomes straightforward and is just the multiplication of the risk factor of the underlying asset and the protocol specific risk factor. As an example, take a protocol (p) with yield bearing assets, which has an *ETH* and a *USDC* vault. The collateral factors of the corresponding $pETH$ and $pUSDC$ tokens are:

$$CF_{pETH}^c = RF_p^c \cdot CF_{ETH}^c \quad (12)$$

$$CF_{pUSDC}^c = RF_p^c \cdot CF_{USDC}^c \quad (13)$$

6.5.2 Maximum Exposure. The maximum exposures give Creditors a tool to set upper boundaries against the overall exposures they can have against certain primary assets or protocols of derived assets. Creditors can even choose to not be exposed to certain primary assets or protocols (via derived assets) by setting the corresponding maximum exposure to 0.

There is again a difference how the maximum exposure is defined for primary assets and for derived assets.

For primary assets, the maximum exposure is defined per primary asset and per creditor. It limits the total amount (in the unit of the underlying asset) of exposure that a single Creditor can have against the primary asset. As such assets with low on-chain liquidity or small market caps can be used as collateral, while market manipulation risks, such as the Mango Market Exploit[12] can be mitigated.

The total exposure of a primary asset takes into account both direct deposits of the primary asset, as indirect exposures via derived assets.

If a maximum exposure is reached, no additional amount of the primary asset can be deposited, nor can any derived asset be deposited that is composed of the primary asset.

For derived assets, there are two mechanisms that provide the Creditor to an upper bound for certain risks. As mentioned before, none of the primary assets of which the derived asset is ultimately composed can exceed their maximum exposure to mitigate risks of market manipulation and low liquidity. Secondly, each creditor can set an upper limit to the total USD exposure of all derived assets of a certain protocol (e.g. the sum of all liquidity position of Uniswap

V2). As such, the Creditor can limit potential losses due to exploits in the protocol.

6.5.3 Minimum Margin. Liquidating unhealthy positions has a considerable gas cost, which liquidators take into account when deciding to liquidate a position or not. This gas cost is independent of the size of the open position. Without a Minimum Margin, this could lead to very small unhealthy positions that are never profitable to liquidate. Or even worse, liquidations that are successfully started, but never profitable to end.

To avoid both problems, accounts must have a Creditor specific minimum value of collateral, the minimum margin M_{min}^c , before they can open a position. The minimum margin can not be used to back liabilities and can be seen as an amount of collateral that is locked to at least cover gas costs of Liquidators.

7 ARCADIA ACCOUNTS

Arcadia Accounts are the main moat of the protocol. Accounts serve as an advanced user-controlled smart wallet from which the owner can perform a multitude of actions: deposit and withdraw assets, use the assets to interact with other DeFi protocols, open a margin account with a Creditor, open positions with the Creditor, change from Creditor or transfer the Account to another owner.

Each Arcadia Account is a separate smart contract, owned by one single user. Users can deploy one or more Accounts through the Arcadia Factory, ensuring the Account adheres to the code as developed by the Arcadia team.

7.1 Margin Accounts

Arcadia Accounts are more than the standard smart contract wallets, they can be used as on-chain Margin Accounts. They are an extension to a smart contract wallet in a similar way how a credit card is an extension to a debit card: The owner of the account/card not only has control over the assets held by the account/card, they can also use the account/card to incur liabilities.

Unlike credit cards however, margin accounts do not rely on trust and reputation to ensure liabilities are paid back.

The Margin Account mitigates the counterparty risk, borne by the Creditor, by guaranteeing that the total value of all assets within the Account is always bigger than the liabilities against the Account (it is over-collateralized):

- It holds the collateral assets and open positions of the Debtor (the Account owner).
- It does the accounting of both the assets and liabilities.
- It enforces the margin requirements continuously:
 - Any Account Action that would bring it at risk of becoming under-collateralized is automatically reverted.
 - If over time, due to changing values of assets/liabilities, it is at risk of becoming under-collateralized, a margin call is triggered.

With Arcadia Accounts, the asset management, Accounting logic, and the enforcement of the margin requirements are 100% implemented on-chain. Arcadia Accounts are permissionless and fully composable with tokens and external DeFi protocols. They can be used by on-chain investors to increase their buying power, to short

assets, or to hold financial contracts such as futures, swaps, options or other derivatives.

In the next paragraphs we will introduce some margin related concepts and express the Margin requirements mathematically.

Note that we do not 100% follow the conventions and terminology of the traditional financial world.

7.1.1 Portfolio Margin. Arcadia Accounts use by default portfolio margin calculations for their risk and margin calculations.

Portfolio margin has a number of advantages over isolated margin (where collateral is not shared between positions at all). It is more capital efficient and prevents unnecessary liquidations or rebalancing of positions, leading to lower fees and potential losses.

Users that still want to have isolated positions can still do so by deploying multiple account and depositing a single asset as collateral in each account.

Portfolio margin is similar to cross-margin in that all assets bought on margin, share the same collateral. But unlike with cross-margin, there is no distinction between realized and unrealized profits. Wins of one position offset losses of other positions, independent if they are realised or not, making Portfolio margin even more capital efficient.

This is because Accounts do not make a distinction between the assets initially deposited as collateral and the assets bought on margin. All assets held by the Account (deposited and bought) are taken in consideration for the margin calculations.

7.1.2 Available Margin. We define the available margin, am , as the total value that the Account can use as margin to secure liabilities.

This total value equals the risk-adjusted portfolio value, expressed by Formula 10, where each asset is discounted with an asset-Creditor specific risk factor (the Collateral Factor).

Hence the available margin equals the Collateral Value as defined in Paragraph 6.4 (we will use the terms 'available margin' and 'Collateral Value' interchangeable).

$$am = v_{coll}^{num} = \frac{\sum_i CF_i^c \cdot v_i^{usd}(a_i) S_{num}}{r_{num \rightarrow usd} 10^{18}} \quad (14)$$

7.1.3 Open position. The open position, op^c , equals the size of the liability that a specific Debtor has with a specific Creditor.

7.1.4 Minimum Margin. As explained in Paragraph 6.5.3, the minimum margin, M_{min}^c , is the minimum amount of collateral value that must be held in an Account to be able to open position with a Creditor.

7.1.5 Used Margin. The used margin, um , is the total amount of collateral value locked by the Account to ensure that the Account remains over-collateralized.

Since collateral in the Account must be held to cover both the open position and the minimum margin, the used margin can be found as:

$$um = M_{min}^c + op^c \quad (15)$$

7.1.6 Free Margin. The free margin, fm , is the remaining amount of collateral value that can be used to increase the open position or that can be withdrawn from the accounts assets.

The free margin can be found by subtracting the used margin from the available margin:

$$fm = am - um \quad (16)$$

7.1.7 Initial Margin. The initial margin of an asset, IM_i^c , is defined in the traditional financial world as the percentage of the value of an asset bought on margin (bought with debt) that must be covered by additional collateral held in the Account at the time of the sale.

As an example, take an asset with an initial margin of 5%, if someone want to buy 10,000 worth of the asset on margin, they need to hold at least and additional 500 worth of collateral in their margin account.

The astute reader will see the similarities with the available margin and Collateral Factors as defined in Paragraph 7.1.2. There is indeed a mathematical relationship between the two.

Take an asset i , for which an amount of x is bought on margin from Creditor C . If a value of x is borrowed, then at least a value of $IM_i^c \cdot x$ must be deposited as collateral into the Account. The account now has an open position of x and holds in total $x + IM_i^c \cdot x$ of the asset i ($IM_i^c \cdot x$ deposited as collateral and x borrowed from Creditor C):

$$\begin{aligned} op^c &= x \\ v_i^{num} &= x + IM_i^c \cdot x \end{aligned} \quad (17)$$

When the minimal amount of collateral is deposited, the free margin will be zero, and after plugging Equation 14 into 16, the used margin equals the collateral value:

$$CF_i^c \cdot v_i^{num} = um \quad (18)$$

And if we ignore the minimum margin (this will be small compared to the open position in most cases) this simplifies to:

$$CF_i^c \cdot v_i^{num} = op^c \quad (19)$$

And finally after plugging in Equations 17, we find the relation between the initial margin and the Collateral Factor:

$$CF_i^c = \frac{1}{1 + IM_i^c} \quad (20)$$

7.1.8 Maintenance Margin. The maintenance margin of an asset, MM_i^c , is similar to the initial margin. It is the percentage of the value of an asset bought on margin, that must be covered by additional collateral held in the Account at all time after the transaction.

If the value of the additional collateral would ever fall below the maintenance margin, a margin call is triggered and a liquidation of the assets of the Account start.

A similar derivation as in Paragraph 7.1.7 can be made for the relation between the maintenance margin of an asset and its Liquidation Factor:

$$LF_i^c = \frac{1}{1 + MM_i^c} \quad (21)$$

7.1.9 Margin Account Health States. There are three different account states, as shown in Figure 3, depending on the Used Margin of the Account and the risk adjusted total values of of the Account.

- (1) **HEALTHY STATE:** The Account is in a healthy state if the used margin is smaller than the Collateral Value:

$$um < v_{coll} \quad (22)$$

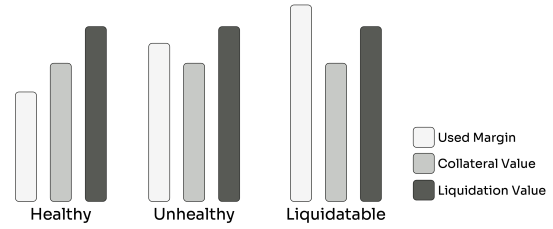


Figure 3: Margin Account health states.

No account actions are restricted for Accounts in a healthy state, as long as the Account is still healthy at the end the action.

- (2) **UNHEALTHY STATE:** The Account is in an unhealthy state if the used margin is smaller than the Collateral Value but bigger than the Liquidation Value:

$$v_{coll} < um < v_{liq} \quad (23)$$

Accounts in the unhealthy state can only be de-risked by or reducing the open position, or adding more collateral assets. It is not possible to withdraw any assets or increase the open position.

- (3) **LIQUIDATABLE STATE:** The Account is in a liquidatable state if the used margin is smaller than the Liquidation Value:

$$v_{liq} < um \quad (24)$$

When an account is in the liquidatable state, anyone can trigger a margin call. The account is now frozen and a liquidation process of the account starts. The liquidation ends when or the account is brought back in a healthy state, or all assets are liquidated.

7.2 Upgradeability

Arcadia Accounts are based on the ERC-1967 standard for Proxy accounts. Each account contract points to a certain Account logic contract. It is the logic contract that implements the features to: deposit/withdraw assets, do flash actions, manage assets, authorise delegations...

Different implementations of the Account logic can (co)exist and each implementation is uniquely identified with a specific version in the Account factory, as illustrated in Figure 4. The Arcadia team can periodically push a new version to the Arcadia Factory, which might add new features, extend logic, or implement other improvements.

Creditors can determine which Account logic versions are allowed to be used by their Debtors. As such, they can block version with certain features or highly-customized Account logic can be implemented for Creditor-specific versions should this be required.

The Account logic is upgradable, enabling existing accounts to make use of newly introduced features in a new Account version if they wish. Upgrading Account logic is always on an opt-in basis, it can never be enforced by the Arcadia protocol. The Accounts can only be upgraded to a different version if both the Account owner and the Creditor (if a margin account is opened) accept the specific version.



Figure 4: Arcadia Proxy Accounts and Account Logic Versions.

When users upgrade their accounts, they don't have to migrate any assets or close liabilities and the accounts keeps the same contract address.

7.3 Multi-asset

The owner of the Account can deposit and withdraw a multitude of collateral assets. The list of assets that can be deposited in Arcadia Accounts is continuously being expanded, thanks to the modular architecture of the Arcadia Registry (see Paragraph 6.1).

By allowing multiple assets to be deposited within a single Account, users, institutions and protocols can build diversified portfolio's and complex strategies.

That does however not mean that Creditors have to be exposed to an ever increasing list of assets. Creditors can limit which assets are allowed as collateral in Accounts of their Debtors, and set appropriate risk parameters for the assets they allow (see Paragraph 6.5).

New assets that can be deposited into Accounts are by default not accepted for a Creditor. Only if the Creditor explicitly allows the asset, can their Debtors use it as collateral.

7.4 Token Standard Agnostic

Accounts are token-standard agnostic, meaning not only typical ERC-20 tokens can be deposited, Also assets complying with other standards, such as ERC-721 ("NFTs", such as Uniswap V3 LP positions) and ERC-1155 assets can be held in an Arcadia Account.

That does however not mean that because ERC-721 is supported, all implementation of ERC-721 are supported. As mentioned in Paragraph 7.3, each individual asset (individual contract address and, if applicable, token Id) must be allowed by the Registry and secondly be accepted by the Creditor before a Debtor can use it as collateral.

Because the Account logic is upgradable, see Paragraph 7.2, even future token standards can be supported by Arcadia Account, if both account owner and creditor agree.

7.5 Composable

Arcadia Accounts themselves are according to the ERC-721 standard and can thus be represented as a single asset on-chain. This means that Accounts are fully composable with existing infrastructure and can be displayed in existing portfolio trackers as such (eg. Zapper, OpenSea, DeBank, ...).

An Arcadia Account can be transferred (or sold!) as a whole, including all assets and liabilities, just like you're used to transferring NFTs.

7.6 Flash Actions

Flash Actions (or optimistic Actions or flash accounting) expands on the concept of flashloans, and are only possible thanks to a unique property of smart contracts: atomicity[13]. Just as with flashloans, each step of the Flash Action must be successful or the transaction as a whole fails.

The final health check of the Account (assets can cover all liabilities) is done at the very end of the transaction. This allows the Account Owner to temporary bring the Account in an undercollateralized (or even non collateralized state) without the risk on bad debt for any Creditor. Since if the Account is not brought back into a healthy state during the transaction, the final health check fails and thanks to atomicity the whole transaction fails.

This gives Account Owners unprecedented flexibility to manage assets and liabilities. In a fully permissionless way they can chain the following actions together (= do a flash action):

- A margin Account can be opened for a new Creditor, if the new Creditor is approved by the Account Owner.
- The Creditor can execute arbitrary logic (e.g. give a flashloan).
- The Account owner can optimistically withdraw assets from the Account.
- The Account owner can transfer assets from his own wallet to the Account or external logic.
- The Account owner can execute arbitrary external logic, where he uses the assets (loaned, withdrawn or transferred) to interact with multiple DeFi protocols to swap, stake, claim...
- The Account owner can deposit recipient tokens back into the Account.

The only requirement is that the Account is in a healthy state at the very end of the flash action.

Flash Actions are a very powerful tool that has no equal in traditional finance. We will list a few examples how flash actions from an Arcadia Account can be used out of the box (the list is far from exhaustive). And keep in mind that all of these actions can be done, even if the assets are used to secure liabilities.

- Rebalance whole portfolio's, swapping a portfolio of n different assets directly to a new portfolio of m different assets.
- Refinance liabilities (change to a different Creditors), without the need to sell any collateral.
- Stake or provide liquidity on approved DeFi protocols. External protocols used in this context will need to provide a receipt token which must be allowed as collateral as well within the Arcadia protocol. Examples can be providing liquidity on Aave (receiving approved aTokens), depositing assets on Yearn (receiving approved yTokens), ...

- Change ranges for Uniswap V3 LP. Contrary to Uni V2 and similar AMMs, Uni V3 positions are meant to be more actively managed in terms of liquidity ranges. Users that deposit Uni V3 positions in their Arcadia Accounts will have the ability to change those liquidity ranges without having to withdraw their tokens first.
- Claim airdrops that depend on address-owned tokens. Arcadia Accounts will feature “flash withdrawals”. This feature can be used by the Account owner to claim airdrops using assets under collateral within their Arcadia Account, without having to close DeFi positions to withdraw their tokens first.
- ...

7.7 Account Abstraction ready

As mentioned in Paragraph 4.2.3, account abstraction is an important step forward in reducing end-user complexities and decreasing operational risks.

At the time of writing this paper, there is no social consensus yet about which standard to use for account abstraction within the Ethereum ecosystem. Hence Arcadia Accounts will initially not be “real abstract accounts” and still require an EOA (Externally Owned Account) owning and controlling the actions of the Arcadia Account. Instead the Arcadia Accounts are “Account Abstraction ready”.

Still some key features of account abstraction are implemented in a non-standardized form. Most notable the ability to bundle multiple actions, and the ability to set an entry point that can do operations on behalf of the account owner.

The Arcadia Accounts are implemented in such a way that if at some point account abstraction is standardized, they can be easily modify to adhere the the agreed upon standard. Even users with existing accounts can (if they want) upgrade their Accounts to this new version with full Account Abstraction, as explained in Paragraph 7.2.

7.8 Intent ready

We truly believe in the benefits intent based architectures bring for end-users, especially for collateral and asset management.

But similarly to what we said in Paragraph 7.7 about Account Abstraction, there is, at the time of writing this paper, no generalized infrastructure or standardization for intents within the Ethereum ecosystem yet.

The Arcadia Accounts are however designed to be “intent ready”. They will initially work with a non-standard single solver for intent based actions, with certain off-chain dependencies.

And secondly, the Arcadia Account require little modifications to support “full intents” as soon as the required infra/standard is available. This would again require an upgrade of the Accounts logic as described in Paragraph 7.2.

7.9 Automated Asset Management

Account owners can appoint one or more Asset Managers, including external smart contracts, which can take actions on behalf of the Account owner.

The Asset Manager is a fully trusted role, and should be used with care. No Account owner should set an Asset Manager which he doesn’t trust with their funds.

The asset manager has multiple use-cases, a few examples are (non exhaustive list):

- Automate strategies via programmatically controlled EOA.
- Automate certain account actions (rebalancing, decrease leverage, stop losses...) with keeper networks.
- Set an Entry point for user operations (abstracted transactions).
- Delegate asset management to third parties.

8 ARCADIA CREDITOR

An Arcadia Creditor is a (set of) smart contract(s), that does the accounting of the open position(s) between one or more Debtor(s) and one or more Creditor(s), and where all open positions are fully secured by the collateral held in Arcadia Accounts, owned by each of the Debtors.

Examples of Creditors can be implementations of peer-to-peer lending protocols, peer-to-pool lending, perpetual futures contracts, options contracts... An Arcadia Creditor can be implemented for basically every financial contract or protocol where a party has, or can have, liabilities.

8.1 Generalised Creditor

Since the Arcadia Financial stack is open and permissionless, every financial contract or protocol with one or more Debtors, can build on top of the Arcadia financial stack.

Protocols that wish to do so, only have to implement the Arcadia Creditor interface and are immediately compatible with the Arcadia Registry and Arcadia Accounts.

Next to the advantages as laid out in Paragraph 4.3, this means they don’t have to implement any logic for collateral management and margin calculations. Greatly reducing smart contract development time, auditing costs and go to market times.

In the next paragraphs we will discuss the most important features of the Arcadia Creditor interface.

8.1.1 Accounting Liabilities. Most important is the correct accounting of the open positions of the Debtors. As explained in Paragraph 7.1.9, the open position is used for the margin calculations to determine the health state of the Arcadia Account of Debtor. Which in turn guarantees that all liabilities can be paid back to the Creditor.

A Creditor must denominate their liabilities in a certain numeraire. This represents the unit of accounting, for example USD, USDC or ETH.

The Arcadia Account of the Debtors will use the same numeraire for the margin calculations.

8.1.2 Risk Management. The owner of a Creditor must set a Risk Manager. The Risk manager can be both an EOA or a smart contract.

The Risk Manager is responsible for setting and maintaining the correct collateral and liquidation factors for all assets that can be used by Debtors as collateral. Next to these factors, the Risk Manager also sets the exposure limits against each individual asset.

In short, it protects the Creditor against market and other risks, and ensures the credit risk borne by the Creditor remains acceptable.

8.1.3 Allowed Account Versions. a Creditor can set one or more Account versions as approved, meaning that any Account with an approved version is able to take liabilities from the Creditor.

8.1.4 Liquidations. Every Creditor must ensure that a suitable liquidation flow is foreseen in case an Account's health state drops to the liquidatable state, as explained in Paragraph 7.1.9.

The Arcadia team already developed a robust liquidation flow which can be re-used by other Creditors, which is further described in Paragraph 8.2.5.

8.2 Arcadia Lending Pools

The Arcadia Lending Pools are just one example of an implementation of an Arcadia Creditor.

Since the Arcadia Lending Pools Arcadia are an implementation of the Arcadia Creditor, they come with the features as mentioned in Paragraph 8.

An Arcadia Lending Pool is peer-to-pool lending protocol, developed by the Arcadia team. On a conceptual level, it consists of a two-sided lending market between liquidity providers and borrowers.

Next to that, the Lending Pools also come with a number of innovations compared to other peer-to-pool lending protocols.

8.2.1 Interests. Arcadia Lending will use a variable interest rate, depending on the utilization rate of the underlying lending pool.

Every-time a user interacts with the Lending Pool (depositing/withdrawing liquidity, taking/repaying debt), the accounting logic for interest payments is triggered and the new interest rate is calculated. Interests are continuously compounding.

The ERC-4626 standard is used to represent both the open positions of Debtors (see Paragraph 8.2.2), as the yield bearing positions of liquidity providers (see Paragraph 8.2.4). Using the ERC-4626 standard makes the accounting logic gas efficient and ensures composability with existing protocols and infrastructure.

8.2.2 Accounting Liabilities. When a Debtor borrows from the Lending Pool, an ERC-4626 token (the debt token or dToken) will be minted to his/her Arcadia Account. The dTokens tokens are not transferable.

The interests on all debt are automatically and continuously accruing via the ERC-4626 standard.

8.2.3 Tranches. Each lending pool consists of one or more risk Tranches. A Tranche is a slice of the total liquidity in the pool, with a different risk-reward profile compared to other Tranches.

For example, the most Junior Tranche (or Insurance Tranche) receives a larger relative share of the interest payments and a larger relative share of the liquidation penalties, but is slashed first should any bad debt occur.

Similarly, a more Senior Tranche may have lower yields through interests and liquidations, but will only be slashed because of bad debt as soon all the more Junior Tranches are completely slashed.

With some game-theory in mind, the overall risk-reward of a more Junior Tranche will be higher than of a more Senior Tranche.

The main advantage of risk Tranches is that Liquidity Providers with different risk/reward preferences can provide liquidity, but that all liquidity is still pooled in the same underlying Lending Pool.

Not fragmenting the liquidity over different pools results in deeper liquidity for the borrowers and more stable interest rates.

8.2.4 Accounting Liquidity. Anyone who owns the underlying asset of the Lending Pool, can deposit liquidity via one of the Tranches.

When a Liquidity Provider (LP) deposits liquidity, they receive an ERC-4626 yield bearing token (ybToken). Each Tranche will have its own ERC-4626 token.

The Lending Pool will do the accounting how much liquidity is owned by each Tranche, while the tranches in turn will do the accounting of how much of that liquidity is owned by their respective LPs.

Yield earned through interests and liquidation fees is divided over the different Tranches (more Junior tranches will earn a bigger relative share of the yield). And via the ERC-4626 standard the yield is further pro-rata distributed to the Tranches LPs.

8.2.5 Liquidations. Liquidation within Arcadia's system is conducted via a three-step process using a Gradual Dutch Auction.

Initially, anyone can initiate the liquidation of an account that falls below the liquidatable health threshold, receiving a reward for doing so.

The auction that follows sees the accounts asset prices gradually decrease until a buyer, typically an MEV searcher looking to minimize slippage, steps in and places a bid. The process allows for partial liquidations to reduce market impact.

Anyone can end the auction, and also receives a reward for doing so, if any if the following conditions are met:

- The Account is back in a healthy state.
- All debt is repaid.
- All collateral assets are sold.
- A certain amount of time (set by the Lending Pool) has passed.

After the auction is successfully ended, the liquidation is settled. There are three different scenarios how a liquidation can be settled, depending on the total proceeds of the auction relative to the debt and liquidation penalties:

- (1) If the auction proceeds are less than the debt, the shortfall is covered by the junior risk tranche's tokens.
- (2) If the auction proceeds exceed the debt but is under the debt plus liquidation penalty, the surplus is shared between Arcadia's treasury and the junior tranche.
- (3) If the auction proceeds surpass the debt and the liquidation penalty cap, excess funds after penalty distribution are returned to the account's original owner.

9 SUMMARY

In conclusion, this paper has explored the development and implementation of the Arcadia Finance Technology Stack: A shared standardized and permissionless layer with battle tested logic for collateral management and margined positions, on top of which financial protocols can build their application specific financial contracts.

By leveraging this technology stack, protocols can streamline their development processes, reducing both time and potential bugs, allowing them to concentrate on refining their core logic.

Simultaneously, users benefit from the simplicity of managing on-chain portfolios through a single account, minimizing risks, and significantly enhancing the overall user experience.

In closing, we invite further exploration and collaboration to unlock the full potential of DeFi and contribute to the ongoing evolution of the finance system.

DISCLAIMER

This document is intended for general informational purposes only and should not be construed as investment advice or a recommendation to buy or sell any investments. The views expressed herein represent the current opinions of the authors and are not presented on behalf of Arcadia Finance, Pragma Labs, or their affiliates. The opinions conveyed do not necessarily align with those of Arcadia Finance, Pragma Labs, their affiliates, or associated individuals, and they may be subject to change without prior updates.

REFERENCES

- [1] H. Adams, M. Salem, N. Zinsmeister, and et al. 2023. Uniswap v4 Core [Draft]. Draft.
- [2] Mathis Gontier Delaunay, Paul Frambot, Quentin Garchery, and Matthieu Lesbre. 2023. Morpho Blue Whitepaper. (2023). <https://github.com/morpho-org/morpho-blue/blob/main/morpho-blue-whitepaper.pdf>
- [3] J. Gregory. 2014. *Central Counterparties: Mandatory Central Clearing and Initial Margin Requirements for OTC Derivatives*. Wiley. <https://books.google.be/books?id=DpXqAwAAQBAJ>
- [4] Martin Hellwig. 2008. The causes of the financial crisis. In *CESifo Forum*, Vol. 9. München: ifo Institut für Wirtschaftsforschung an der Universität München, 12–21.
- [5] Euler Labs. 2023. Protocols vs Products. (2023). <https://www.euler.finance/blog/protocols-vs-products>
- [6] Robert Leshner and Geoffrey Hayes. 2019. Compound: The money market protocol. *White Paper* (2019).
- [7] P. Millett. 2002. *Lending and Borrowing in Ancient Athens*. Cambridge University Press. <https://books.google.be/books?id=0IriOXSe5QC>
- [8] Fabian Schär. 2021. Decentralized finance: On blockchain-and smart contract-based financial markets. *FRB of St. Louis Review* (2021).
- [9] M. Simmons. 2019. *Collateral Management: A Guide to Mitigating Counterparty Risk*. Wiley. <https://books.google.be/books?id=DMOKDwAAQBAJ>
- [10] Maker Team. 2017. The dai stablecoin system. (2017). <https://makerdao.com/whitepaper/DaiDec17WP.pdf>
- [11] Harry Thornburg. 2020. Aave protocol whitepaper v1. (2020). https://github.com/aave/aave-protocol/blob/master/docs/Aave_Protocol_Whitepaper_v1_0.pdf
- [12] Fran Velasquez. [n. d.]. DeFi Exchange Mango's \$114M Exploit Was 'Market Manipulation,' Not a Hack, Ex-FBI Special Agent Says — coindesk.com. <https://www.coindesk.com/tech/2022/10/20/defi-exchange-mangos-114m-exploit-was-market-manipulation-not-a-hack-ex-fbi-special-agent-says/>. [Accessed 30-01-2024].
- [13] Yining Xie, Xin Kang, Tieyan Li, Cheng-Kang Chu, and Haiguang Wang. 2022. Towards Secure and Trustworthy Flash Loans: A Blockchain-Based Trust Management Approach. In *International Conference on Network and System Security*. Springer, 499–513.