The Arcadia Finance Technology Stack:
A standardized on-chain architecture for collateral management [Draft]

February 2024

Thomas Smets
Pragma Labs
Brussels, Belgium
thomas@pragmalabs.dev

Jasper Van Pee
Pragma Labs
Brussels, Belgium
jasper@pragmalabs.dev

Zeki Ulu
Pragma Labs
Brussels, Belgium
zeki@pragmalabs.dev

ABSTRACT

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

KEYWORDS

Collateral Management, DeFi, Margin, EVM

1 INTRODUCTION

This paper will explain the rationale behind the Arcadia Finance Technology Stack, what its main benefits are, and how it is technically implemented (the latter starts from Section 5 onwards).

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 of three protocols, each with distinct responsibilities and permissions:

- The Arcadia Registry.
- The Arcadia Accounts.
- The Generalized Creditor.

To establish a shared understanding, this paper starts with Section 2, where concepts and terminology related to collateral management and margined positions are defined.

Section 3 provides a brief exploration of the use of collateral in the traditional financial system.

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

In Section 5 our implementation of such a novel architecture is introduced: the Arcadia Financial Technology Stack.

Finally, each of the underlying protocols of the Arcadia Financial Technology Stack is 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: The terms Debtors and Creditors are often used throughout this paper. Our definition is broader than the commonly accepted context within traditional debt arrangements. The term Debtor is used 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...). 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 IN TRADITIONAL FINANCE

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.

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 digitalization 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].

But collateralization in the traditional financial world is not without its problems. The 2008 financial crisis highlights 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 centralisation 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.

4 COLLATERAL MANAGEMENT IN DEFI

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 in 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 (Decentralized Applications), combined with a shared state between all those applications, make different financial applications and assets composable and inter-operable by default.

Hence it is not a coincidence that some of the first blockchain applications with product market fit, like Maker[10], Aave[11] or Compound[6], were collateralized lending protocols.

4.1 General Principles

Lending protocols are not the only DeFi protocols that use collateralization to mitigate counterparty risks between different user-groups in some way or shape. Perpetual protocols, Prime brokerage protocols, option protocols etc. all require some or all users to deposit collateral.

Most of these DeFi protocols require their users to over-collateralize their open positions, meaning that the sum of all collateralized assets must always be larger, with a certain safety margin, than the sum of all liabilities.

For example, if users want to borrow $800 worth of token A via a certain lending protocol, they must first deposit and lock at least $1000 worth of token B as collateral in said protocol.

Additionally, most protocols have some kind of liquidation mechanism in place to prevent bad debt when the market moves against a user's position. Taking the previous example, if token B drops

in value compared to token A, say from $1000 to $900, the user's position can be (fully or partially) liquidated.

4.2 Shortcomings and Inefficiencies

There are however still a number of shortcomings and inefficiencies which need to be overcome, before DeFi can be broadly adopted as the go-to infrastructure for collateral management.

4.2.1 Collateralized in Name Only. In 2022 DeFi experienced its own financial crisis. In a few months time, many of the ecosystem's key (albeit mostly centralized) players, and a number of protocols collapsed. 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.

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

4.2.2 Hacks and Exploits. A major problem in DeFi today is the number of hacks and exploits occurring. The estimated amount of funds lost in 2023 due to exploits 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 specific factor contributing significantly to this 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 redundant, 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.

4.2.3 Fragmented and Isolated Collateral. Having a global shared state across applications and assets is often cited as one of the key advantages of blockchain over centralized financial infrastructure[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.

But if we look at the reality today, we must admit 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, and the average DeFi-user has assets fragmented and isolated across a multitude of protocols.

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

- As mentioned in Section 4.2.2, there is no shared collateral management layer, each protocol with collateral has their own non-standardized implementation.
- Protocols are build around a single specific asset type (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.

While isolated margin positions have their benefits (they isolate risks), they are not capital efficient. The volatility of a portfolio of assets is always equal to, or lower than the weighted volatility of the individual assets (assets losing in value can be compensated with assets increasing in value).

Having a lower volatility, reduces the number of rebalancing actions (and their fees), reduces the number of liquidations and depending on the correlations between collateralized assets, the Creditor might use less strict margin requirement.

Fragmentation of assets not only results in capital inefficiencies, it also contributes to a challenging user experience, more on that in Section 4.2.4.

4.2.4 End User Complexity. End-users do not "want" to manage collateral. Rather, they want to optimize portfolios to achieve certain objectives (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 abstracted away. Important note, abstracting away technical complexities should never come at a cost of hiding risks or relying on centralized and custodial solutions (as is too often the case).

As mentioned in the previous Section 4.2.3, the bad user experience is partly due to the fragmentation of assets and positions. Having a standardized collateral management layer with standardized dapps across different protocols would benefit all users.

Rebalancing portfolios often require multiple transactions per asset and per position. For which the user today has to execute multiple sequential transactions per asset, instead a single transaction that rebalances the complete portfolio.

Lets 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, they wants to diversify risks over 5 different LST service providers. Assuming our user has wrapped Ethereum, they 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.

Another 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 portfolios 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 Decentralized 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 decentralized 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
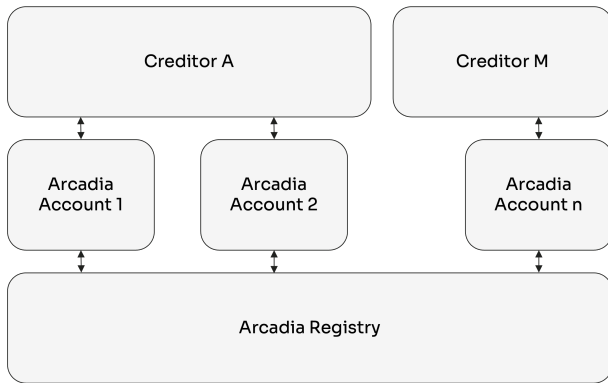
Figure 1: The Arcadia Financial Technology Stack.

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 timelines. 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 DEXs (Decentralized Exchanges), without the need and risks of redundantly implementing the complex maths related to concentrated liquidity.

Over-collateralized protocols would benefit from a similar architecture as well. 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. Fast moving, end-user facing features can be 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 of three protocols:

- The Arcadia Registry.
- The Arcadia Accounts.
- The Generalized Creditor.

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

Arcadia Accounts owners can batch together multiple account actions, automate account actions and interact optimistically with third party protocols. Arcadia Accounts enable an unprecedented user experience, without compromising on self-custody and composability.

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

Lastly, the Generalized 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 Finance Technology Stack.

Protocols 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 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 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.
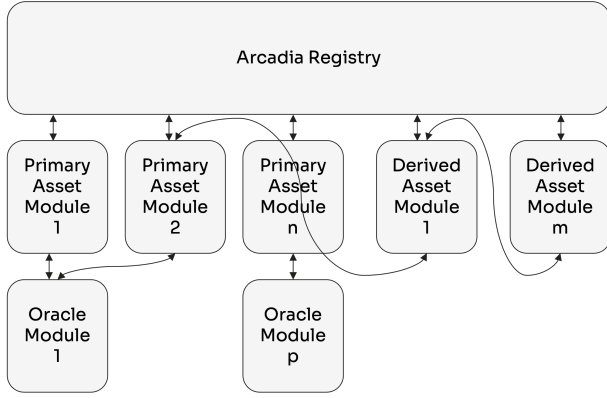
Figure 2: The Arcadia Registry and Modules.

- 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. 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 $r_{ba \to qa}$, returned by the Oracle Module.

$$r_{ba \to qa} = r_{oracle} \frac{10^{18}}{S_{oracle}} \tag{1}$$

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. Similar to how 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 $r_{ba \to qa}$ 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 differently. When the asset is the base-asset of the oracle:

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

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

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

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

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

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

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}.

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

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 $v^{usd}(a_u)$ from the Registry, using the underlying asset amounts a_u from Equation 6.

$$v^{usd}(a_u) = v^{usd}(f_u(a_{asset})) \tag{7}$$

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.

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

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:

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

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 portfolio values, where assets are discounted with an 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:

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

There are two important risk adjusted portfolio 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 Factor LF_i^c. 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:

$$CF_i^c = f(\forall CF_u^c, RF_p^c) \tag{11}$$

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:

$$CF_{pETH}^c = RF_p^c \cdot CF_{ETH}^c \tag{12}$$

$$CF_{pUSDC}^c = RF_p^c \cdot CF_{USDC}^c \tag{13}$$

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 of, 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 V3). 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 largely 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 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 source 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 to 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). The Margin Account will:

- Hold the collateral assets and open positions of the Debtor (the Account owner).
- Do the accounting of both the assets and liabilities.
- Enforce 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 portfolio margin calculations by default 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 Accounts 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 realized 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).

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

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:

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

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 Account's assets.

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

$$fm := am - um \qquad (16)$$

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 an 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
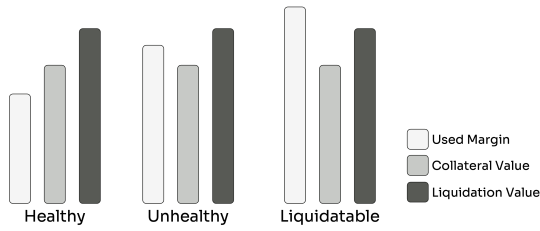
Figure 3: Margin Account health states.

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):

$$op^c = x$$
$$v_i^{num} = x + IM_i^c \cdot x \qquad (17)$$

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:

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

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

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

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

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

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:

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

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:

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

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:

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

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 asset or to increase the open position.

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

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

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 either the Account is brought back in a healthy state, or all assets are liquidated.

7.2 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 portfolios 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.

7.3 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 implementations of ERC-721 are supported. As mentioned in Paragraph 7.2, 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 7.4, even future token standards can be supported by Arcadia Accounts, if both the Account owner and Creditor agree.

7.4 Upgradeability

Arcadia Accounts are based on the ERC-1967 standard for proxy contracts. 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,

Figure 4: Arcadia Proxy Accounts and Account Logic Versions.

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.

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

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 are 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 ("Can the assets cover all liabilities?") is done at the very end of the transaction. This allows the Account Owner to temporary bring the Account in an under-collateralized (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 in one 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 they 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). Keep in mind that all of these actions can be done, even if the assets in the Account are used to secure liabilities.

- Rebalance whole portfolios, 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 4.2.4, 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 modified to adhere to 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 7.4.

7.8 Intent Ready

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

However, as of this writing, no generalized infrastructure or standardization for intents exist within the Ethereum ecosystem.

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

Arcadia Account require little modifications to support "full intents". Thus as soon as the required infrastructure/standard is available, it can be implemented. This would require an upgrade of the Accounts logic as described in Paragraph 7.4.

7.9 Automated Asset Management

Account owners can appoint one or more Asset Managers, including external smart contracts, which can take actions on behave 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 they 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, escrow services... An Arcadia Creditor can be implemented for basically every financial contract or protocol where a party has, or can have, liabilities.

8.1 Generalized Creditor

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

Protocols that whish 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 themselves. This greatly reduces 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. In turn, this 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. As explained in paragraph 7.4, multiple Account versions can coexist.

Each Creditor can approve one or more of these Account versions, meaning that only Accounts with an approved version are able to open positions with the respective 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 are an implementation of the Arcadia Creditor, they come with the features as mentioned in Paragraph 8.

An Arcadia Lending Pool is a 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), and 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 the Debtor's 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 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 due to bad debt as soon as each more Junior Tranche is 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 to keep track of 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. A schematic of an Arcadia Lending Pool with Tranches is illustrated in Figure 5.

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

8.2.5 Liquidations. Liquidations for Arcadia Lending Pool are 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.
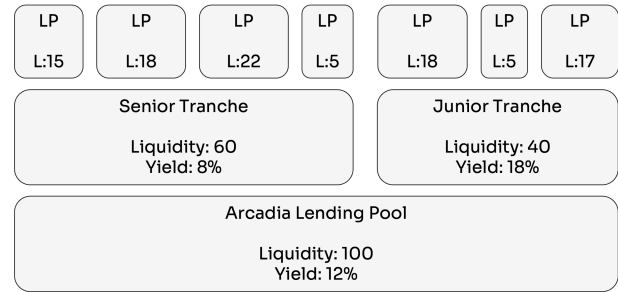

Figure 5: Arcadia Lending Pool with two Tranches.

Figure 6: Liquidation Settlement scenarios.

After the liquidation is initiated, a Gradual Dutch Auction starts where the Account's asset prices gradually decrease until a buyer, typically an MEV searcher, steps in and places an atomic 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, as illustrated in Figure 6, 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 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 most 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.

Another innovative feature is that the Account Owner, or third-party automation tools, can still intervene after an auction is initiated. After sufficient debt is repaid, the auction can be ended with minimal loss for the original owner. This approach offers account holders a proactive means to manage and potentially reduce the impact of liquidation events, ensuring a more dynamic and responsive liquidation process.

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 Arcadia 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 financial 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=0IriOXQSe5QC

[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.