Ithil

The Web3 Wizard

Ithil is a financial interoperability layer that connects the whole web3 space facilitating new value creation via crowdlending.

V1.0.1 - November 22, 2022

Abstract

Ithil aims to bring decentralised services to the web3 space via a well-thought system of undercollateralised loans - a game-changer for traders, liquidity providers, and other protocols who can now rely on various capital-efficient products to address their needs.

Modular and easily upgradable, Ithil offers users and other protocols easy interactions with the web3 space as a whole, enabling an entirely new range of financial opportunities. Liquidity is taken from liquidity providers (or Lenders), who can stake any whitelisted ERC20 token and get a high APY on that same token getting exposure to the ecosytem with a reduced risk. In fact, their capital is spread over several protocols and it is protected by an efficient liquidation system.

Contents

1 Introduction

In the current Web3 space there are several opportunities, from DeFi high yields to NFTs and gaming, with new ones coming up almost every day. Users are shown a wide landscape to interact with, depending on their risk appetite and personal ideas. However, all the subspaces are set apart one another, and end up being self-contained bubbles that don't interact with each other: DeFi, NFTs, gaming and metaverse are all separate from each other and the composability of one another is still uncharted territory.

While for the lenders, the typical parameters LPs look for are a good APY, the market exposure (such as by holding volatile tokens), the underlying protocol's reliability and security, the liquidity available or TVL, their personal preferences and several other factors difficult to predict or model. With Ithil they are presented a novel opportunity to get exposed to the high yields on the whole Web3 space in a combined way, reducing the overall risk as well as increasing the return rates on their deposits.

1.1 What is Ithil

At its core, Ithil is a protocol that combines the liquidity providers' deposits to other protocols to create novel financial services for its users. In this way, Ithil can be considered as a *trait-d'union* between the many protocols existing in the space, users and liquidity providers.

1.2 What can be done on Ithil

With Ithil, anyone can

1. Become **liquidity provider** (LP) by staking their tokens to get a solid APY. A extensive choice of whitelisted tokens can be staked in Ithil, and the APY generated is in the same token: DAI stakers will get DAI tokens, SHIB stakers will get SHIB tokens, and so on. In this way, Ithil offers an attractive staking opportunity, also for holders of volatile tokens.

2. Optimise their investments by placing some collateral in Ithil and then using the LP liquidity to optimise the capital allocation. Thanks to an internal system of undercollateralised loans (**the Internal Lending Engine**), the total capital invested can be much higher than the collateral placed: by placing only 100 DAI worth of collateral, a user can perform an investment worth 1000 DAI or more.

3. Be a **liquidator** by constantly checking the losing positions and liquidating them in a fully on-chain way, thus getting high rewards.

4. Similarly to the liquidators, a user can become a **harvester** by automating the strategies that require a farming process, getting a part of the harvest.

5. Join the **community** by buying and holding Ithil's governance token, using it in the liquidation process or participating in governance voting. Ithil's innovative backing system ensures that Ithil's value mathematically increases in time, thus encouraging holders to commit to the community for a long time.

1.3 Ithil's unique features

Many of Ithil's features are rarely found in the current web3 landscape. Some of the differences are the following:

- **Modularity** allows Ithil to list or de-list virtually any service with a governance vote, hence, allowing Ithil to continuously be updated with the most recent protocols, always offering what is best and trending to the users, be it DeFi, Metaverse, Play to Earn or anything.
- **High capital efficiency** over-collateralisation has always been an essential aspect in DeFi loans, which reduces the possibility for users to boost their available funds.[1] Thanks to a novel lending model, Ithil makes *under-collateralised* loans possible, thus allowing a completely new set of on-chain financial services.
- **Opportunity to stake virtually any token**: few protocols offer a single-sided APY on ideally any token, usually being restricted to stablecoins, big cap tokens like WBTC and WETH or the protocol native token. Ithil's lending vaults are instead token-agnostic: any ERC20 token can be whitelisted and staked and it generates APY in the same token.
- **Real-yield sustainable APY** even on stablecoins' vaults, thanks to sustainable treasury management and a non-dilutive boosting system.
- **Capital protection** for liquidity providers: an efficient liquidation model and an algorithmically-maintained insurance reserve protect the LPs' liquidity, thus assuring long-lasting earnings with limited risk exposure.
- **Solid tokenomics** which mathematically assures Ithil's governance token minimum exchange value to constantly increase over time, or more precisely, to never go below an always increasing floor, thus rewarding holders committing to the protocol for a long time. See 6.3 for details.

1.4 Ithil's mission

Commitment to offering an easy and sustainable set of financial products and help users get the best out of them is the main focus of Ithil.

By interconnecting the several protocols existing in the subareas of web3, Ithil can create innovative investing opportunities (DeFi strategies), help users get what they want (on-chain mortgages) or simply enjoy the space better (access gated communities by renting NFTs).

Through capital boosting, Ithil wants to mitigate the intrinsic advantage of wealthy individuals, allowing small users to fully embrace the opportunities web3 has to offer.

Building trust is of primary importance, especially in an ecosystem where scammers, ponzi schemes, and ill-modelled backing systems are unfortunately too popular. Ithil commits to giving real value to the community: every surplus earned by the protocol is algoritmically injected into the backing contract, thus increasing the value of its token and the financial power of its treasury in a clear and sustainable way.

[1] Or makes this process very costly, mainly through the so-called *folding strategies* on money markets like Aave.

2 Core concepts

This section summarises the inner workings of the protocol, which will be treated more in-depth in the following sections.

2.1 The Vaults

The **Vaults** are an ERC4626 smart contracts that collects liquidity to be used in services required by the users. The liquidity contained in each Vault is made of a **whitelisted ERC20 token**, which can be in principle any tokens, from stablecoins to meme and rebasing tokens. The entire liquidity is conceptually split into three parts:

1. **Liquidity Providers' Liquidity** (LPL). Anyone staking ERC20 tokens inside a Vault is considered an LP. Such liquidity can be lent to any of the Services contracts; it collects fees, can be redeemed any time [2] by unstaking it, and is protected by the IRL and by liquidators (it has a positive APY).

2. **Insurance Reserve Liquidity** (IRL). This is a part of the liquidity bootstrapped by fees and maintained by the governance. It cannot be lent to the Services. It collects fees, it cannot be redeemed by anyone [3] and is not protected: it is used to protect the LPL in case a Bad Liquidation Event (see Section 2.6) occurs.

3. **Boosting Liquidity** (BL). A lump sum injected into a Vault by a *booster* to bootstrap the TVL and increase LPL's APY. In principle, anybody can be a booster, but the typical use cases are Ithil treasury (eventuall coming from the backing contract) and external DAOs. As all liquidity inside a Vault, it can be lent to the Services but it does not collect fees (they are redistributed to the LPL), is protected by IRL and liquidators and can be redeemed only by the booster itself.

The liquidity is used by lending it to the Services (see 2.2) in the form of direct non-collateralised loans (see 2.3). Only Services can take loans from a specific Vault and repay them: in a Vault's smart contract, the `borrow` and `repay` functions are only callable by specific whitelisted addresses. This is made to ensure the assets never go to a random address and makes safe internal loans possible.

Besides managing staking and unstaking of LP tokens, a Vault is also responsible for registering the total issued loans, whitelisting Services, computing the interest rate (see 2.4) of each loan and calculating the payoff to give to the borrower.

2.2 Services

The **Services** are smart contracts that fully use the inner composability of web3 to offer a specific service on-chain. Possibilities include DeFi investments, NFT

[2] Assuming it is not lent in that particular moment, although it collects high fees if it is. See Section 3.5 for more details.

[3] But if it is too high, the governance can withdraw it to rebalance it. See Section 6.5 for more details.

purchases, mortgages and fx forwards on commodities. The contract usualy takes a margin from the user, a loan from the desired token Vault and calls an external protocol to obtain some asset(s).

Ithil implements a modular structure for the Services, which are totally different one-another and just need to implement an `open` and `close` functions. When the external protocol is called, the Service uses Ithil LPs liquidity, eventually together with the user margin (see 2.3) to obtain some **assets**, which are then locked into the Service contract itself and become **held tokens**. The original tokens taken from Ithil to obtain the desired assets are referred as **owed tokens**, in that the Service *owes* them to Ithil in order to repay the loan.

The workflow is summarised as follows:

- The user transfers some ERC20 tokens as **margin** to the desired Service contract and chooses the amount of liquidity to be taken from the Vault (usually the frontend precomputes the best amount automatically).
- If the amount is within the established limits (see 2.4), the Service takes the necessary liquidity from the Vault of the chosen token.
- The Service calls the relevant external protocols, collects the assets received from them and registers the open position on-chain. Notice that *these assets do not go to the original user*: they stay locked in the Service contract. Without this system, it would be impossible to grant an undercollateralised loan (see 2.3).
- The position is part of the Service's state and it is visible on-chain as a NFT. It can be traded by the user or by Ithil (see **??**. The `quote` public function of the Service allows checking the value of the held tokens with respect to the owed tokens. If a position's value decreases below a specific **risk factor** of the margin, the position is **liquidable** and can be closed by liquidators (see 2.6).
- At any time, the user can close the position. In this way, the Service will call the relevant external protocols to exchange its held assets into owed tokens, then transfer them to the Vault to repay the loan. The difference between the obtained amount and the repayment due to the Vault represents the user's profit.

This workflow is totally abstract and can be applied to a wide variety of Services. Each time a new feature is implemented, it must be **whitelisted** by the governance to be able to take loans from the Vaults. Only whitelisted addresses can call the borrow and repay functions of the Vaults and ensure the loan is repaid at the end.

2.3 Undercollateralised loans

One of the most critical parts of the system is borrowing liquidity from the Vaults and using it to offer a service. Since the liquidity taken may be higher than the margin given by the user, this has the same effect of a **loan** issued to implement something directly with the Vaults' liquidity.

An important takeaway is the fact that *this does not lend directly to the user*: Ithil does not transfer any liquidity or asset to the user wallet until the

position is fully closed and the loan repaid. Instead, it is an internal loan, "from Ithil to Ithil", though any eventual profit is transferred to the user at the loan repayment time.

In order to access a service, the user has to transfer some **margin** to Ithil. This margin covers the position from eventual losses caused by unfavorable market movements. The margin, just like the assets coming from a given protocol, is locked in the Service contract until the position is closed or liquidated. It can be seen as a "payment" (which is fully refunded if the position is closed profitably) necessary to use some of the Vault liquidity.

Once the position is opened, and the assets are inside the Service, the part of assets coming from the user's margin is called the **collateral** placed for the loan. The ratio between the collateral [4] and the loan taken is called the **collateralization** of the loan: in the limit of zero liquidity borrowed from the Vault (no leverage), the collateralisation is infinite. It decreases as the loan increases with respect to the margin amount. The loan is called **over-collateralised** if its collateralisation is higher than 1: the loan taken is lower than the margin placed, so the position is virtually riskless. [5] If the collateralization is lower than 1, we say the loan is **undercollateralised**.

Thanks to the system described, there is no need to distinguish between the two types of loans: Ithil allows in principle any value of collateralisation. However, maximum loan value is placed to put a cap on the riskiness of a single position (see 2.4).

2.4 Interest rate

When the Service borrows some liquidity, the target Vault registers the loan in an on-chain state and applies a tailored **interest rate** to that particular loan. The details on the calculation are given in 4.2. The interest rate is an essential part of the fees generated by Ithil, which contribute to the Vault's APY, and represent the compensation given to the LPs for having their liquidity locked in a Service.

The interest rate is **dynamically computed**. More precisely, the interest rate is

1. directly proportional to the **leverage**,
2. directly proportional to the **vault usage** (how many loans are open at the moment),
3. directly proportional to the **risk factor** of the chosen service (see 2.6),
4. inversely proportional to the **insurance reserve** available (see 2.5).

When a position is open, the interest rate is attached to that position onchain: when the position is closed or edited, the interest rate is used to calculate the due amount to the Vault.

The interest rate cannot exceed a **maximum interest rate**, decided by the governance: this is Ithil's way to control the riskiness of the investments. Instead of maximum leverage or maximum usage, all the risk data are collected

[4] Or the part of the assets coming from the collateral: see 4.2 for more details.

[5] Although, in a cross-margin situation, impermanent loss can negatively affect the Vault's liquidity in this case as well, thus liquidators are still necessary. See 4.2 for more details.

together, thus leading to dynamic limits for all such data. In particular, the **maximum leverage** allowed depends on the particular Service (risk factor) used, on the usage, and on the insurance reserve available at that moment.

The dynamic interest rate also automatically allows the Vault to adjust its parameters based on the usage. If the liquidity is mostly idle, the interest rate will be very low, thus encouraging users to enter into an investment. Conversely, if the vault is heavily used, or if many LPs have redeemed their liquidity (see 3.5), users will be discouraged to take further loans due to a high-interest rate. As the positions are closed, more liquidity will become available and entering into new positions will be interesting again. More examples and remarks are given in Section 3.

2.5 The insurance reserve

Part of the Vault's balance cannot be borrowed and is used as a guarantee for possible losses caused by BLE (see 2.6): this is the **insurance reserve**, a fundamental part of Ithil's Vault.

An algorithmically computed part is given to the insurance reserve when fees are generated. The precise calculation is done in 3.3, but in general, the smaller the insurance reserve balance, the higher the portion of fees goes to the insurance reserve. The minimum is attained when the ratio between the insurance reserve and the total balance (loans included) reaches an **optimal ratio**. This is a token-specific percentage that prescribes the "best" portion of liquidity to be allocated as insurance: more balance would mean too low an interest rate, thus affecting the APY negatively, while less balance would mean too high an interest rate, thus affecting usage negatively.

When the actual ratio is higher than the optimal ratio, the governance can choose to rebalance it by withdrawing some or all of the difference. Notice that this can only be done to make the ratio closer to the optimal one: the governance *cannot* withdraw on a ratio smaller than optimal.

When a BLE occurs (see 2.6), the liquidity of the insurance reserve is used to compensate the liquidator and to cover the eventual Vault's losses caused by the BLE.

2.6 Liquidation

Since the Vault's liquidity is used to perform investments, an unfavorable market movement could put the loan taken at risk. This is why **liquidations** are crucial for the health of the protocol.

Liquidations are managed on-chain and in a decentralised way: anyone can liquidate a given position, as far as the position is **liquidable**, i.e. has a value that is too close to the original loan to be kept open.

The typical liquidation process is summarised as follows.

- A liquidator checks whether a position's quoted value (given by the Strategy-specific `quote` function) has gone below the minimum value given by the risk factor (see 5.2).
- If it is the case, the liquidator can liquidate the position using one of the available liquidation methods (see 5.3): if Ithil's on-chain check confirms the liquidability of the position, the position is closed.

- The closure proceeds are transferred to the Vault to repay the loan and relative fees. At the same time, the remaining part is split between the Vault and the liquidator as compensation for having liquidated the position (see 5.4 for more details on this distribution).

- In the case the proceeds are insufficient to repay the loan, we call it a **Bad Liquidation Event (BLE)**. In this case, the Insurance Reserve (see 2.5) is used to repay the missing part of the loan and to compensate the liquidator.

Since liquidating a position gives the liquidator a riskless gain, this creates "liquidation battles" in which liquidators compete for the fastest liquidation.

In order to obtain their rewards, liquidators have to stake governance tokens: the more one has staked, the larger the rewards will be (see **??**).

2.7 Tokenomics

Ithil's **governance token** ITHIL has the primary function of helping the protocol grow and improve the services it provides to its users. ITHIL token holders are rewarded in many ways, depending on their usage of the token. Moreover, a simple and innovative backing system ensures its value never fall below a continuously increasing floor.

There are three main utilities in ITHIL token:

- **Liquidations**. ITHIL holders can stake their token into Ithil's liquidation contract to get higher rewards.

- **Voting**. Each ITHIL staked in the governance contract represents one vote: staking many ITHIL makes one's vote to count more.

- **Redeem**. ITHIL can be redeemed on the protocol any time at the **backed price**. See 6.3 for more details.

The protocol's proceeds above the ones necessary to keep a sound APY are transferred by the governance into a **backing contract**, thus increasing the backed price, protecting the holders from market fluctuations and indirectly rewarding holders for committing to the protocol for a long time.

3 Liquidity provisioning

As mentioned in 2.1, Ithil's liquidity is obtained from liquidity providers, who stake their tokens, making them available to Ithil's Strategies, and by governance injections, used to boost the APY for LP's and protect their capital.

In order to deposit an ERC20 token on the Vault, such token must be **whitelisted** by the governance. This system avoids malicious tokens, such as the ones with blocked transfers, to be injected into Ithil. The governance can also decide to *remove* a given token from the whitelist: in this way, unstaking of the token is always possible, as well as closing positions on that token, but staking and opening new positions will not be possible.

3.1 The role of LPs

Liquidity providers (LP) are the owners of what has been called LPL, standing for Liquidity Providers' Liquidity. This is distinguished from the other two types of liquidity: Insurance Reserve Liquidity or IRL, and Boosting Liquidity or BL. Except for boosters, anyone staking their ERC20 tokens on Ithil's Vault is considered a LP, and is entitled to the fees generated by the protocol on the deposited token (see 3.3).

Example 1. If an LP has staked 1000 DAI on the Vault, and afterwards a position that generates fees in DAI (for example, a DAI leveraged staking, see 4.4.1) is closed, part of the fees generated will be claimable by the LP, who will be able to withdraw more than 1000 DAI. If fees are generated in any other token, they are not delivered to the LP mentioned above, but will go to the other token's respective LPs.

Example 2. If an LP has staked 1000000 SHIB, and afterwards a position which generates fees in SHIB (for example, a short trade on SHIB, see 4.4.2) is closed, part of the fees generated will be claimable by the LP, who will be able to withdraw more than 1000000 SHIB. If fees are generated in any other token, they are not delivered to the aforementioned LP.

LPs are essential to Ithil, since the protocol usage would be much less interesting without their liquidity. Two opposite forces control the eventual APY observed by the LPs.

1. If a small amount of LPL is in the Vault, the relative portion of BL is higher. Thus a small usage is compensated by a high boosted APY, which attracts new LPs (see 3.6 for more details).

2. If a large amount of LPL is in the Vault, the cost of making riskier investments is lower (see 3.3). Thus a lower BL portion is compensated by a higher APY generated by fees.

Eventually, the amount of LPL staked in the Vault will reach a dynamic equilibrium, based on the usage and market conditions: below that equilibrium, BL-generated APY will attract new LPs; above that equilibrium, LPs may look for other investment opportunities.

3.2 Wrapped tokens

In order to register the liquidity staked, and the amount to be given back to traders, the Vault transfers **wrapped tokens** to LPs at the moment of the staking. Their names are obtained by prepending an `i` in front of the native token's symbol: the wrapped token for `TKN` is `iTKN`.

The wrapped tokens are minted when an LP stakes, and burned when unstakes native tokens. In order to calculate the number of tokens to be minted and transferred to the LP, we use the following formula:

$$M_{\text{iTKN}} = \frac{S_{\text{iTKN}}}{LPL_{\text{TKN}}} D_{\text{TKN}}, \tag{1}$$

where M_{iTKN} is the quantity of wrapped tokens `iTKN` to be minted, S_{iTKN} is the total supply of `iTKN`, LPL_{TKN} is the LPL denominated in `TKN`, and and D_{TKN} is the quantity of `TKN` the LP has deposited.

In order to compute LPL_{TKN} in Solidity, we can use the following equation, which reflects the conceptual division of the Vault's liquidity made in 2.1.

$$LPL_{\text{TKN}} + IRL_{\text{TKN}} + BL_{\text{TKN}} = B_{\text{TKN}} - Lock_{\text{TKN}} + L_{\text{TKN}}. \tag{2}$$

Here B_{TKN} is the available balance of `TKN` inside the vault, L_{TKN} is the quantity of `TKN` taken as loans (see 4.2), and $Lock_{\text{TKN}}$ is the fees still locked in the Vault (see 3.4). Since the balance is obtainable by the ERC20 method `balanceOf`, we see that the amount of data to be registered on-chain is four, with the fifth one obtainable by Equation (2).

Remark 1. In the particular situation, of inception, where $LPL_{\text{TKN}} = 0$ and $S_{\text{iTKN}} = 0$, we define the backing as 1-to-1: for 1 `TKN`, the Vault transfers 1 `iTKN`.

The quantity

$$P_{\text{iTKN}} = \frac{LPL_{\text{TKN}}}{S_{\text{iTKN}}}, \tag{3}$$

is called the **share price** of the wrapped token `iTKN`, and it can be considered as the amount of `TKN` to stake in order to obtain one unit of `iTKN`. A positive APY corresponds to an increasing share price: the faster it increases, the higher the APY.

A fundamental property of the share price is the fact that it does not change when users stake their tokens. Indeed, assuming one LP deposits an amount D of `TKN`, then the amount of `iTKN` to be minted is calculated by Equation (1). Note that we drop the subscripts to ease the notation. We then have

$$P' = \frac{LPL + D}{S + \frac{S}{LPL} D} = \frac{LPL}{S} = P.$$

In Section 3.5, we show in the same way that the share price does not change when users unstake their tokens, thus preventing arbitrage on the wrapped token's share price.

3.3 Fee generation

The expression (3) for the share price makes evident that, in order to increase the share price, and thus the APY, the LPL must increase. This is where **fees** come into play.

3.3.1 Interest rate

When some amount of tokens `TKN` are taken from a Strategy as a loan (see 4.3), the user who launched the investment needs to pay some *interest rate* and a *fixed fee* on the transaction.

The interest rate is calculated with the following formula (we avoid the subscripts `TKN` for readability, but recall that all data, in particular the interest rate, are token-specific):

$$r_{\text{nd}} = \frac{1+\delta}{2\kappa}\left(r_{\text{base}} + \frac{L + \max(L - IRL; 0)}{B + L - IRL}\beta\right). \tag{4}$$

As in Section 3.2, L are the total loans already taken, IRL is the Insurance Reserve Liquidity as in Section 2.1, and B is simply the total vault's token balance (regardless of LPL, BL and locked fees). We also call $B - IRL$ the *free liquidity*, i.e. the liquidity ready to be used for the investments. The new parameters in Equation (4) are the following.

- The **collateralization** κ, which is the ratio between the margin posted as a collateral by the user and the amount taken for the loan (see 4.2). In the particular case of zero loan taken, i.e. $\kappa = \infty$, this makes the interest rate zero.
- The **risk factor** β, which is the riskiness of the chosen investment. This has key importance in liquidations (see 5.2), and has both a governance and an algorithmic component.
- The **discount** δ, defined as the ratio of the initial asset balance in the strategy, and the final one. This achieves *diversification*: it will be more costly to obtain the same interest rate in a intensively used strategy, with respect to a poorly used one.
- The **base rate** r_{base}, which is the minimal interest rate to be applied. The governance chooses it.

3.3.2 Fees calculation

The interest rate is computed by the vault at the moment the position is opened and the loan is taken, and is attached to the position in that moment. The fees to be paid to the vault are then calculated *at the closure* of the position, using the following formula

$$f = m(f_{\text{fixed}} + rT), \tag{5}$$

where

- m is the total investment amount, composed by margin *and* loan taken,
- f_{fixed} is the fixed fee, decided by the governance and token-specific,
- r is the interest rate computed in (4),
- T is the time passed between the opening and closure of the position.

If a loan of l was necessary to open a position, an amount of $l + f$ must be given back to the Vault at the closure.

3.3.3 LPL/IRL split

When fees are generated, they are distributed between LPL and IRL following the **insurance portion** ι, which is algorithmically computed in 3.3.4 from the Vault's state. Precisely, we have

$$\begin{cases} LPL' = LPL + (1-\iota)f \\ IRL' = IRL + \iota f. \end{cases} \tag{6}$$

In particular *when fees are generated, the share price* (3) *increases*, thus allowing the LPs to redeem more than the amount they initially staked.

Notice that, thanks to Equation (1), only fees generated *after* a given LP has staked contribute to that LP gain, and once the fees are generated, they can be redeemed at any time by that given LP. This is due to the fact that withdrawals do not change the share price (see 3.2). In this way, faster redeemers do not get more fees, but rather fewer, since they renounce to the subsequent fees generated after they unstaked.

3.3.4 Calculation of ι and the optimal ratio

As can be seen by (6), the Insurance Reserve Liquidity IRL increases at each fee generation. The quantity ι is computed to make the IRL increase faster when its amount, compared to the total Vault's liquidity, is low. Explicitly, we have

$$\iota = \left(1 - \frac{\max(IRL - L; 0)}{B}\right)\rho_0, \tag{7}$$

where

- IRL is the Insurance Reserve Liquidity,
- L are the loans taken,
- B is the total token balance of the Vault,
- ρ_0 is the **optimal ratio** of the Vault.

The optimal ratio prescribes the "best" proportion of the Insurance Reserve with respect to the total Vault's balance. It is computed algorithmically, assuming that the more the loans are at risk, the higher ρ_0 should be. Calling β_l the risk factor for each loan (see 5.2 for more information about risk factors), and l the amount of such loan, then we have

$$\rho_0 = \frac{1}{L}\sum_{l \in \text{loans}} l\beta_l. \tag{8}$$

The optimal ratio and IRL adjustments have a crucial role in Ithil's tokenomics: see 6.2 for more details.

3.4 Locking

When fees are generated, the part of fees not included in the insurance reserve is *locked* and it cannot be immediately withdrawn. Unlocking occurs linearly with time, and after 6 hours the entire amount of fees is available to be withdrawn. This system is classical among many DeFi protocols and it is necessary to mitigate attacks in which an attacker stakes a huge amount of capital just before fees are generated (eventually through a flashloan) and unstakes it in the same block, thus claiming the majority of the generated fees.

Technically, this is achieved as follows: every time fees are generated, an onchain datum `latestRepay` is updated with the current `block.timestamp`, and another onchain datum `currentProfits` is updated as the current *locked* profits plus the generated fees (minus insurance reserve):

$$\text{Current Profits} = \text{Old Locked Profits} + \text{Fees} - \text{IR Portion}.$$

At any given time, the locked profits (the missing summand in Equation (2)) are calculated as

$$Lock = \max\left\{\text{currentProfits} \cdot \left(1 - \frac{\text{timetamp} - \text{latestRepay}}{21600}\right); 0\right\}.$$

The locked profits are then subtracted from the current liquidity to calculate the amount to deliver to the LP when unstaking (see Section 3.5).

3.5 Redemptions

The wrapped tokens `iTKN` can be redeemed at any time to obtain `TKN`. The amount obtained is calculated by solving Equation (1):

$$R_{\text{TKN}} = \frac{LPL_{\text{TKN}}}{S_{\text{iTKN}}} B_{\text{iTKN}} = P_{\text{TKN}} B_{\text{iTKN}}, \tag{9}$$

where R_{TKN} is the quantity of TKN which are redeemed (transferred to the burner), B_{iTKN} is the amount of `iTKN` burnt by the Vault, and the other data are as in Section 3.2. Notice that the amount redeemed is simply computed using the share price P_{TKN} defined in (3).

Similarly to what we showed in Section 3.2, we show that redemptions do not modify the share price. Dropping subscripts, if one burns B `iTKN`, afterwards we have

$$P' = \frac{LPL - \frac{LPL}{S}B}{S - B} = \frac{LPL}{S} = P.$$

In short, neither redemptions nor deposits modify the share price, but only fee generation as in 3.3. In particular, this means that one cannot grab the fees already generated by staking and unstaking tokens and that unstaking first does not provide any advantage in terms of redeemed amount: arbitraging on the share price change is not possible.

Remark 2. Since `iTKN` and `TKN` are linked to one another via the share price, any external market [6] willing to price the pair `iTKN/TKN` must stick with the share price. Otherwise, arbitrage on the external market would be possible (in one direction or the other, depending on whether the market price is higher or lower than the share price).

[6] We are thinking of external dexes like Uniswap

3.5.1 Lack of liquidity

In the case too much liquidity is locked into the Strategies, and at the same time a great amount of LPs want to redeem their `iTKN` simultaneously, the Vault may not have enough liquidity to immediately redeem them.

In this case, the first to redeem will get their `TKN` immediately, but then the denominator of Equation (4) will drastically decrease. Therefore, a user willing to open a new position in this situation will need to pay a *very high interest rate*, thus discouraging the user from taking a further loan. As soon as the open positions are closed, more and more liquidity becomes available (and at a higher share price, due to fees generation), thus also the redeemers who did not manage to redeem their `iTKN` will now be able to do that.

Interest rate and the liquidation system (see Section 5) assure that the positions are either liquidated after a long time, in which the interest rate has eroded too much of the investment's value, or accumulated so large fees, that it is very profitable to keep them staked.

3.6 Boosting liquidity and APY increase

The Boosting Liquidity (BL) already mentioned in Section 2.1 is particularly important to bootstrap the protocol and can increase the Vaults' APY when they fail to be competitive.

Anybody, at any time, can inject or withdraw some BL in a given Vault (i.e. in a given token). Strategies can then borrow this liquidity and therefore generate fees. However *these fees are distributed to the LPL and IRL*: in other words, the BL does not accumulate fees.

In order to see this, we see the system (6), which tells that the total fees f are not distributed to the BL. Moreover, Equation (2) shows that when BL is deposited or withdrawn, neither LPL nor IRL change, since the amount $B - BL$ remains constant.

Since the BL has an APY of zero, the typical booster are Ithil's treasury and backing contract (in stablecoins) or external DAOs in the context of partnerships (in any token).

The BL is particularly efficient in the case the LPL is very low: in this case, the **APY boost** experience by early stakers is so high, that it can attract stakers and therefore increase the LPL and improve Ithil's liquidity.

Example 3. Assume the BL in a given Vault is of 1000000 `DAI`, while just 1 `DAI` has been staked, so that the LPL is 1 `DAI`. Assuming that, in a given day, a small gain of 0.01% has been made thanks to the fees generated by the loans taken from the BL, and assuming a ι of 25%, this means that 75 `DAI` are distributed to the LPL, which has made a gain of 7500% in only one day, thus experiencing a 2737500% APR on `DAI`. [7]

The enormous APY obtained as described in Example 3 will then benefit early stakers, who will compete to have the higher portion of the BL's fees. Notice, however, that *the booster's liquidity is not given out to LPs*: if a booster puts some amount M as BL, then the same amount M can be withdrawn by the same booster at any time. Moreover, booster liquidity is the *most senior* when

[7] The APY, in this case, is very similar since the compounding effects are minimal.

it comes to Bad Liquidation Events (see Section 5.4.1): in this way, a lower risk compensates for the lack of APY.

4 Strategies

Ithil's **Strategies** are smart contracts representing a particular investment, that is, an exchange of user's and Vault's liquidity to obtain some **assets** on external DeFi protocols. [8] Users, in order to launch a Strategy, need to deposit some **margin** into the Strategy itself. Closing a position amounts to exchanging the assets to obtain the original token. This one is given to restore the Vault's liquidity (plus fees) and to pay the user in the case the investment has proven profitable.

Ithil's modularity allows for many possible strategies, with new ones being listed and old ones delisted by the governance, following market conditions and sentiments. The architecture in place allows for immediate and secure integration of new strategies.

4.1 The Base Strategy

In order to provide a standard for all strategies to be implemented, Ithil provides a **Base Strategy** (BS) contract. All Strategies must inherit from this contract in order to be listed. The BS is an `abstract` Solidity smart contract, which only implements the generic Strategy logic with respect to loans, repayments, and collateral management. In particular, it does not implement the interaction with external protocols, but leaves the relevant functions as `virtual`. The only thing specific Strategies must do is implement the BS's `virtual` functions, thus simplifying a lot the developers' task of building new strategies.

In this section, we consider a Strategy that exchanges `TKA` for `TKB` on some external protocol in *some way* (recall that the Base Strategy abstracts the actual implementation: we will discuss this in 4.3). In Ithil's terminology, `TKA` is the **owed token**, in that the Strategy *owes* it to the Vault after borrowing it, and `TKB` is the **held token**, since it is held (locked) within the Strategy.

4.1.1 Opening a position

We describe here in detail the workflow of the BS, i.e. the sequence of calls and methods used by the BS to open a position.

1. The user posts some amount m of **margin** into the desired Strategy (in particular, the user also has to choose an implementation: see 4.3). The margin can be in either `TKA` or `TKB`.

2. The user chooses a quantity A and an **amount** to be invested. This is the quantity of `TKA` to be exchanged in order to get `TKB`.

3. If the user has posted the margin in `TKA`, the Strategy takes $A - m$ `TKA` from the Vault as a loan. Otherwise the margin is in `TKB` and the Strategy takes A `TKA` as a loan from the Vault.

[8] Internal transfers can also be taken into consideration.

4. The Strategy computes the **collateral** placed by the user for the loan. If the margin is in `TKA`, then the collateral is equal to the margin, while if it is in `TKB`, the `quote` function of the Strategy's implementation (see 4.3) calculates the value of the margin in terms of `TKA`.

5. The Vault computes the interest rate (see 3.3.1), which is passed to the Strategy as a return value.

6. The external contract required by the specific strategy's implementation is called (see 4.3), and the amount of `TKB` obtained is registered.

7. All the relevant data are registered, and a `Position` data structure is saved on-chain in the Strategy's state.

In this way, the position is open onchain, as it is part of the Strategy's state, and can be read by anyone. The assets received, i.e. the `TKB` obtained from the external protocols and eventually from the user's margin, are locked in the Strategy. Notice that this allows for an under-collateralised loan from the Vault in a secure way (the user cannot run away with the assets).

4.1.2 Closing a position

When a position is opened by a user, its address is stored as the `owner` member of the `Position` structure. That address can **close** the position it opened by launching the specific Strategy's function. Again, the particular actions to perform the closure belong to the Strategy's implementation (see 4.3). However, in the Base Strategy the general workflow of starting with `TKB`, obtaining back `TKA`, and repaying the Vault and the user is implemented.

1. The user calls for the closure of the position. If the caller is the position's owner, the closure starts.

2. If the user has posted, at the opening, a `TKA` margin, the entirety of `TKB` obtained during the Strategy's opening [9] are exchanged through the external DeFi protocol (see 4.3) to obtain back as many `TKA` as possible. Otherwise, it exchanges only the necessary amount of `TKB` to obtain enough `TKA` to repay the loan taken from the Vault plus fees.

3. In normal conditions, the liquidation system assures the quantity of `TKA` obtained is sufficient to repay the Vault and its fees. If it is not the case, the closure is reverted: only liquidators can close the position in this case. See Section 5 for more details about Ithil's liquidation system.

4. The Strategy repays the Vault by transferring the necessary amount of `TKA` in it. The remaining part of `TKA` or `TKB`, depending on the user's initial margin, are transferred back to the user. All the necessary states involved in the fees generation process (see 3.3) are updated.

5. The `Position` data structure corresponding to the position just closed is deleted.

[9] If TKB supports rebasing, the reflections are also considered

In a profitable investment with a `TKA` margin, the amount of `TKA` obtained at the closure is higher than those spent at the opening. In the case of margin in `TKB`, the amount of `TKB` necessary to close the position is lower than the ones obtained at the opening. In both cases, the user experience a gain on a capital amount that is much higher than the margin posted, thanks to the funds borrowed from the Vault.

4.1.3 Editing a position

While a position is open, its owner can decide at any moment to **edit** it by posting extra collateral. This can be useful if the position is too close to a liquidation (see Section 5), and is known in TradFi as *margin call*. [10] The collateral is transferred to the Vault if it is in the owed token, and to the Strategy if it is in the held token.

When this is done, the due fees are computed and registered in the `Position` struct, and the interest rate is re-computed. In this case, also, the fees are paid at the position's closure.

4.2 Loans and repayments

The Base Strategy also deals with Vault's loans and repayments. It assures that all loans are correctly repaid and that the due fees are generated so that this core feature is respected regardless of the particular Strategy's implementation.

The two Vault functions to deal with this process are `borrow` and `repay`. Such functions have the `onlyStrategy` modifier, meaning that they can only be called by **whitelisted** addresses, i.e. the Strategies which are inserted in the Strategy list by the governance.

- `borrow` When this function is called, the Strategy passes the risk factor (see 5.2), the collateral placed, and the desired amount to be borrowed as parameters. Then the Vault computes the loan's interest rate, transfers the tokens to the Strategy, and registers the loan taken onchain. If the free liquidity is not enough, or if the interest rate exceeds the maximum one (this happens, for instance, if the required leverage is too high: see 3.3.1), the transaction reverts.

- `repay` This function, also callable by the Strategy only, withdraws the owed tokens from the Strategy, updates the Vault's net loans (a state variable) and repays the position's owner by transferring the extra amount, not necessary to repay the loan and fees. Finally, it generates fees as described in 3.3.

4.3 Strategy implementation

The Base Strategy does not deal with the actual implementation of any particular investment. By "implementation", we mean all the necessary external contract calls [11] which have the final effect of having an *exposure* to some external market movement. Such movement can be, in principle, of any type: from

[10] If the position is liquidable, liquidators can perform a forced margin call and become the position's owners. See 5.3 for more details.

[11] Of course, even internal transfers are *a priori* possible.

the increase of the share price of an LP token (for instance when staking into a yield-providing protocol 4.4.1), to the appreciation of one token with respect to the other (for instance in margin trading, see 4.4.2), to gain exposure to real-world assets like in synthetics-related strategies, to raffles, et cetera.

Ithil's modularity allows to list or de-list Strategies at any time, through a governance decision. Since much of the logic contained in the Base Strategy, the effort necessary to code a particular implementation is reduced to the minimum.

The Base Strategy has three `virtual` functions: two `internal` ones named `_openPosition` and `_closePosition`, and a `public` one named `quote`. The Strategy's implementation just needs to inherit the Base Strategy and implement these functions.

- `_openPosition` accepts an `Order` data structure, which has `spentToken` and `obtainedToken` as address members. The goal of `_openPosition` is to spend `spentTokens` on some external DeFi protocol, and obtain some quantity of `obtainedToken`. Also, the amount spent should not be above the `maxSpent` field of the Order, and the amount obtained not below the `minObtained` field of the order. Except for that, the function will attempt to obtain as many tokens as possible, or spending as few as possible depending on the particular cases. This is an `internal` function.

- `_closePosition` accepts a `Position` data structure, which has `owedToken` and `heldToken` as address members, and an `allowance` member representing the quantity of `heldToken` entitled to the position. The goal of `_closePosition` is to spend `heldToken` on some external DeFi protocols, to obtain some `owedToken`. The maximum amount of `heldToken` to spend is always the position's `allowance`. [12] Depending on the cases, the function will try to spend as little as possible to obtain a fixed amount, or to spend the maximum amount to obtain as much as possible. This is an `internal` function.

- `quote` accepts two tokens `source` and `destination`, and an `amount` as parameters. The goal of this function is to tell how many `destination` tokens one would obtain by exchanging an `amount` of `source` tokens with the same methods as the one implemented in `_closePosition`. This is a `public view` function.

Once these three functions are implemented, the Strategy contract is not `abstract` anymore, and it can be deployed and, eventually, whitelisted by the governance to be used on Ithil. If a Strategy is implemented, deployed but not whitelisted, it will not be able to call the `borrow` and `repay` functions of the Vault, thus resulting in a revert. Notice that since `_openPosition` and `_closePosition` are `internal`, they cannot be accessed directly. There is no way to exchange tokens within a Strategy except by calling a function in the Base Strategy contract.

4.4 Example of strategies

In what follows, two examples of strategies are given. It is essential to understand that these two are just examples, and they are inserted here *for illustration*

[12] Except for tokens with a reflection: in this case the reflected tokens can also be spent.

purposes only: they are by no means the "most important" or "most used", or "most anything" ones. Moreover, giving a comprehensive treatment of these strategies, or of others, is out of the scope of this whitepaper.

However, since they are somewhat different in nature, riskiness, and implementation, they illustrate well the flexibility of Ithil's architecture.

4.4.1 Leveraged yield farming

In **Leveraged Yield Farming** (LYF), a token of the Vault's can be staked in an external protocol to obtain the APY provided by that protocol and eventually some other tokens associated with liquidity mining programs. Since there are numerous protocols in the DeFi space offering this service, this Strategy is, in reality, a *class of strategies*: Strategies using different protocols must be implemented separately.

Typically, these kinds of strategies are of relatively low risk. However, the possibility of negative APY on the external protocol, amplified by the eventual leverage requested by the user, make it necessary to provide a non-zero risk factor (see 5.2) to these strategies as well. Among other things, elements such as the external protocol's reliability, its TVL, the nature of the staked tokens and the presence of liquidity mining programs are considered when assessing the riskiness of a staking strategy.

In what follows, consider a scenario in which `TKA` can be staked into some external **staking protocol** which we call SP. For simplicity, assume there is no liquidity mining program, and that SP gives back wrapped tokens `spTKA` to represent the staked liquidity. As a disclaimer, notice that the numbers listed in the following example are fictitious.

- A user posts a margin of 100 `TKA` to the LYF Strategy contract and decides to go with a 10x leverage.
- The Strategy will borrow 900 `TKA` from the Vault and stake the 1000 `TKA` on SP, obtaining say 1000 `spTKA`.
- Assume that, after one month, the rolling APR of SP has been 60%, thus giving a monthly gain of 5%. Also, assume that the monthly interest rate applied by Ithil's Vault has been 3%.
- If the user closes its position, the Strategy will redeem 1000 `spTKA` on SP, to obtain 1050 `TKA`. The Strategy then repays the Vault with 927 `TKA` (900 `TKA` borrowed, plus 3%).
- The remaining 123 `TKA` are given back to the user.

Notice that the user has seen a 23% gain in one month, using a staking protocol that only gives 5% of yield. This is the effect of the leverage, damped by Ithil's interest rate.

Notice also, that if in the previous scenario the monthly gain obtained by SP were *below* Ithil's interest rate of 3%, then by closing the user would incur a *loss* or, if the loss is too heavy, the user could be liquidated (see Section 5).

The margin, together with the liquidation process, ensures that the 900 `TKA` lent by the Vault to Strategy, will be repaid in any case to the Vault (except BLE's, see 2.6, in which the IRL guarantees the repayment).

4.4.2 Margin trading

The **Margin Trading Strategy** (MTS) consists of exchanging directly some amount of `TKA` into some other amount of `TKB` using an external Decentralised Exchange (*dex*) or Automatic Market Maker (*AMM*). [13] As in 4.4.1, since many protocols are offering these services, this Strategy is a class of Strategies: Strategies using different protocols must be implemented separately.

Depending on the tokens exchanged, these Strategies can be highly hazardous: if the price of `TKB` moves in a way, which is different from the one foreseen by the user at the moment the position is opened, the user can incur high losses, amplified by the leverage requested. Conversely, the potential gains are incredibly high if the price moves favourably for the user, thanks to the leverage. The risk factors (see 5.2) of these strategies are specific to the pair of tokens traded, and elements like historic volatility of the tokens' price and the external dex's pool liquidity are taken into account to assess the riskiness of a token pair.

Let us make a numerical example of a **long position** of `TKA` against `TKB` (the user foresees that the exchange rate `TKA/TKB` will *increase*) on an external dex that we call DEX:

- A user posts a margin of 100 `TKA` to the MTS contract and decides to go long with a x10 leverage on `TKB`.
- Assuming an exchange rate of 50 `TKA/TKB`, the MTS will then borrow 900 `TKA` from the Vault, say at a daily interest rate of 0.4% and exchange 1000 `TKA` to obtain 20 `TKB` from DEX.
- If the user closes the position after 10 days, and the exchange rate provided by DEX has increased by 20% to 60 `TKA/TKB`, then exchanging 20 `TKB` back will provide 1200 `TKA` to the MTS.
- The MTS will repay the 936 `TKA` to the vault (900 borrowed plus 4% interest) and deliver the remaining 264 `TKA` to the user.

Notice that, with a market movement of only 20%, the user has realised a 164% gain from the 10x leverage. In the case of a **short position** of `TKA` against `TKB` (the user foresees that the exchange rate `TKA/TKB` will *decrease*) the situation is similar:

- A user posts a margin of 100 `TKA` to the MTS contract and decides to go short with a x10 leverage on `TKB`.
- Assuming an exchange rate of 50 `TKA/TKB`, the MTS will then borrow 20 `TKB` from the vault, say at a daily interest rate of 0.4% and exchange them on DEX to obtain 1000 `TKA`. At this point, 1100 `TKA` are locked in the MTS as allowance.
- If the user closes the position after 10 days, when the exchange rate has decreased by 20% to 40 `TKA/TKB`, then only 832 `TKA` are necessary to obtain the 20.8 `TKB` back to repay the vault (20 borrowed plus 4% interest).

[13] Many times, the terms are used quite interchangeably, and some protocols have aspects of both a dex and an AMM, therefore the distinction is often not so important.

- The Strategy performs the swap of 832 `TKA` to repay the Vault, and delivers the remaining 268 `TKA` to the user.

Again, a market movement of only 20% has made the trader earn 168% from the 10x leverage.

5 Risk and liquidations

Since Ithil's loans to a given Strategy can be under-collateralised, if the market moves in an unfavourable direction, the liquidity borrowed from the Vault during a given position can become at risk. If this happens, the position can be **liquidated**, i.e. forcefully closed in some of the available ways (see 5.3), in order to complete the repayment of the loan taken from the Vault.

Ithil's liquidation system is totally *decentralised*: anyone can liquidate open positions by launching the functions of the `Liquidator` smart contract. The check for liquidability of a given position is totally *on-chain*. It is based on the usage of the `quote` function of the particular Strategy (see 4.3): this assesses the value of a given position in the moment the function is called, and together with the **risk factor** contributes to the calculation of the **liquidation score** (see 5.2). If the liquidation score of a given position is positive, the position is **liquidable**, and can be closed by the liquidator, who gets rewards doing this (see 5.4).

Liquidation is done at the level of the Base Strategy (see 4.1), in particular, it does not depend on the particular strategy implementation: the liquidator is free to choose the most convenient liquidation system for the particular strategy.

5.1 Liquidators

In what follows, a **liquidator** is any address attempting to liquidate one or more positions, using any of the available liquidation systems (see 5.3). In order to liquidate, the liquidator must give

- The **Strategy's address**: the particular Strategy in which the liquidated position resides.
- The **position's id** of the liquidated position: recall that the positions are stored onchain, and the id is required to identify them.
- Further parameters, depending on the particular liquidation system.

Once the position is liquidated, it is deleted and it will not be visible anymore onchain. Since liquidating a position can come with high rewards (see 5.4), a fast liquidation is crucial for the liquidator, in order not to be front-run by other liquidators.

When one of the available liquidation systems is used, the contract checks the liquidability of the position on-chain. If the position is found to be not liquidable (with a negative liquidation score: see 5.2), the call is reverted. In particular, although not necessary, an off-chain check would be beneficial to any liquidator to avoid the gas costs associated with a reverted call. The critical function `computeLiquidationScore` (see 5.2) is `public view` and can therefore

be called without any gas cost, thus assuring complete alignment of on-chain and off-chain checks.

Ithil's **treasury** (see 6.5) participates in the liquidation process just like any other liquidator: no particular advantage is given to the treasury with respect to any other address. The rewards eventually accumulated by the treasury are totally redistributed to Ithil's community via the backing contract (see 6.3), thus contributing to the increase of Ithil's value.

5.2 Risk factor and liquidation score

The **risk factor** is a parameter already discussed upon in Section 3.3: it is a number $0 < \beta < 1$ assessing the riskiness of the particular investment taken. The risk factor is governance based and it is both strategy and token-specific. Notice that, following 3.3.1, the higher the risk factor, the higher the interest rate necessary to open a position.

The risk factor enters directly into the calculation of the **liquidation score** λ, which is the parameter used to assess the liquidability of a given position. The liquidation score is computed via `computeLiquidationScore`, which is a `public view` function. Such function computes the **profit-and-loss** P&L of a given position using the `quote` function. Precisely,

$$\text{P\&L} = \begin{cases} \text{obtained} - \text{principal} - \text{fees}, & \text{if collateral} = \text{owed}, \\ \text{allowance} - \text{cost}, & \text{if collateral} = \text{held}. \end{cases} \tag{10}$$

In (10), "obtained" is the number of owed tokens obtained by exchanging the allowance, and "cost" is the cost of repaying the quantity principal + fees; both are calculated by `quote`. Thus they are as if the position had to be closed when the P&L is calculated. In short, the P&L is the number of tokens the user would obtain if the position were closed at that moment: see Section 4 for more details on these concepts.

The liquidation score is finally computed as

$$\lambda = \beta \times \text{collateral} - \text{P\&L}. \tag{11}$$

The position is said to be **liquidable** if $\lambda > 0$: as mentioned, if a liquidation call results in $\lambda \leq 0$, the call is reverted and liquidation does not occur.

Notice that, if $\lambda = 0$, we have $\text{P\&L} = \beta \times \text{collateral}$, therefore β can be seen as the *loss* the collateral can experience before a position gets liquidated. In particular, a positive collateral amount is still left at the moment liquidation occurs, thus making the Vault's repayment more robust and allowing to reward the liquidator (see 5.4).

5.3 Types of liquidations

There are three types of possible liquidation procedures, all of them have the primary goal of either repaying the Vault's loan, or making the position's liquidation score (11) negative by increasing the collateral.

5.3.1 Forced closure

In this method, the position is closed by launching a `_closePosition`. This has the same effect, on the Vault's side, as the user directly closes the position, except that the payoff given to the user will be significantly smaller due to the liquidator's reward. As in 4.1.2, the owed tokens obtained by the closure are transferred to the Vault in order to repay the loan, and the remaining collateral is split between the liquidator and the treasury (see **??**).

This liquidation system has the lowest risk for the liquidator since no amount needs to be paid. However, it is more complex (and thus expensive) in terms of gas.

Example 4. Let us take the example of the Margin Trading Strategy as in 4.4.2. Assume the risk factor of the position is $\beta = 0.5$, that the position is long `TKB` versus `TKA` with 100 `TKA` as collateral, the exchange rate is 50 `TKA`/`TKB`, and assume the user has requested x10 leverage so that 1000 `TKA` are exchanged to get 20 `TKB` from an external DEX. If the exchange rate goes down until 48 `TKA`/`TKB`, and the fees accumulated so far are of 15`TKA` the quoter would calculate

$$\text{P\&L} = 48\ \texttt{TKA/TKB} \times 20\texttt{TKB} - 900\texttt{TKA} - 15\texttt{TKA} = 45\texttt{TKA},$$

therefore the liquidation score would be

$$\lambda = 0.5 \times 100\texttt{TKA} - 45\texttt{TKA} = 5\texttt{TKA} > 0.$$

Therefore, the position can be liquidated by swapping back the allowance of 20`TKB`, obtaining 960`TKA` from the DEX which will be used to pay the 915`TKA` due to repay the Vault's loan and fees, and the remaining part is split between the treasury, the liquidator, and the original position owner following the scheme of Section 5.4.

5.3.2 Position purchase

The liquidator can also purchase the position's held tokens by repaying the Vault. The liquidator transfers the principal amount and the fees to the Vault, and the Strategy transfers the tokens locked in the position to the liquidator's address. The position is then deleted, and the liquidator can choose what to do with the assets obtained.

This is the quickest and most gas-efficient way to liquidate, but it requires a high down payment from the liquidator (the principal and due fees to the Vault). Moreover, the assets obtained a value at risk for the liquidator, who must manage them well afterwards to obtain a good gain.

Example 5. In the same situation as Example 4, a liquidator can directly transfer 915`TKA` to the Vault to close the liquidable position, and have the allowance of 20`TKB` be transferred to the address used to call the liquidation function. At this point, the liquidator is free to use these 20`TKB` as preferred. Swapping them back to DEX as in 4 (if the price hasn't changed in the meantime) would yield 960`TKA` to the liquidator, thus realizing a gain of $960 - 915 = 45$ `TKA` for the entire procedure. However, the liquidator might want to keep them for other reasons, or use them in separate investments.

5.3.3 Margin call

Not technically a "liquidation", since the effect of the margin call is not the closure of a position, a liquidator can choose to insert more margin into the position to make the liquidation score lower than 0 (see 5.2). If this is the case, the position *stays open* and the liquidator becomes its owner, thus leaving the liquidator with the decision of what to do with it.

It requires a small down payment and is not too expensive in terms of gas, but keeping the position open the liquidator stays exposed to the position's risk, and must be careful not to be in turn liquidated if the market keeps on moving unfavorably.

Example 6. In the situation of Example 4, the liquidator can choose to perform a margin call by transferring 20`TKA` to the vault. In this way, the position's principal becomes 880 `TKA`, thus the new payoff will be

$$\text{P\&L} = 48\ \texttt{TKA/TKB} \times 20\texttt{TKB} - 880\texttt{TKA} - 15\texttt{TKA} = 65\texttt{TKA}$$

and the new collateral is $100 + 20 = 120\texttt{TKA}$. Therefore, the liquidation score becomes

$$\lambda = 0.5 \times 120\texttt{TKA} - 65\texttt{TKA} = -5\texttt{TKA} < 0$$

and the position is not liquidable anymore. The liquidator becomes now the owner of this open position, and can choose what to do with it. Closing it immediately would deliver a payoff of 65`TKA` to the liquidator, which will then realize a gain of $65 - 20 = 45\texttt{TKA}$ for the full procedure. Notice that in all cases, the liquidator gets 45`TKA` in the situation considered: all methods give the same payoff opportunity to the liquidator.

5.4 Liquidator's rewards

We assumed the liquidator gets all the rewards to simplify the treatment in the previous examples. In reality, the reward given to the liquidator is split between the Insurance Reserve (see 2.5) and the liquidator. The liquidator can increase the amount of reward obtained by staking governance tokens (see **??**).

The minimum reward a liquidator gets is adjusted to the various liquidation systems in the following way.

- In a **forced closure** liquidation, the liquidator gets the greatest between the trader's P&L and 5% of the borrowed assets obtained by closing the position.
- In a **position purchase** liquidation, the liquidator can purchase the assets at a 5% discount with respect to the market fair value, obtained by Ithil's `quote` function.
- In a **margin call** liquidation, the liquidator can reduce the due fees by 50%.

All these numbers refer to the *maximum* reward a liquidator can obtain. The actual reward depends on how much governance token the liquidator has staked into the Liquidator's contract. The governance decides a *maximum stake* M, which can change over time depending on the governance token's value: the

liquidator's reward will then be a factor $0 < \gamma < 1$ of the maximum reward, computed as

$$\gamma = \max\left\{\frac{S}{M}; 1\right\} \tag{12}$$

where S is the amount of governance tokens staked by the liquidator.

5.4.1 Bad liquidation events

A **bad liquidation** is a liquidation in which P&L < 0, with notation as in (10) or, looking at the definition of the liquidation score in (11),

$$\lambda > \beta \cdot \text{collateral}.$$

In this situation, the amount obtained by closing the position is not sufficient to repay the principal and interest of the position (or the cost to repay the loan is higher than the allowance in the case of a held collateral).

Due to a BLE, part of the Vault's liquidity is lost, which is the only case where this happens. From Section 5.4, we see that in this case the liquidator's reward is in any case lower than in a good liquidation event: liquidating quickly, when the liquidation score is low, is encouraged through a higher reward.

In the case of a BLE, the IRL covers for the liquidity loss (see 2.5), including the funds necessary to reward the liquidator. In this way, the LPs do not see their APY go negative even if there is a BLE, as far as there is enough IRL. If a BLE occurs *and* the IRL is not enough, then the LPL decreases.

Remark 3. Notice that the minimum reward to give to a liquidator is necessary, even in the case of a BLE. Indeed, the gas cost related to a liquidation could bring the liquidator to wrap the call into an `if` statement, which does not perform the call if the reward is too low. In this case, a position that is losing too much could stay open and keep on losing virtually until the entire allowance is worthless, thus making the Vault lose a significant amount of funds. With the minimum reward, it is ensured that it is always convenient to liquidate a position, and as discussed in this Section, the sooner it is liquidated, the better.

6 Tokenomics

Ithil's tokenomics is conceived to maximize the protocol's attractiveness by raising its APY, consolidating the earnings of the LPs, and rewarding the community solidly and sustainably.

6.1 The governance token

The governance token ITHIL represents a unit portion of the protocol: any owner of ITHIL can be considered a member of Ithil's community. ITHIL token has the following characteristic.

- It is *deflationary*: the total supply of ITHIL is minted at the launch, and no other tokens are minted after that moment.
- It is a *utility token*: it can be used in various ways on the protocol in exchange for services.

- It is *backed*: it can always be exchanged on Ithil's backing system (see 6.3) at a never decreasing floor price.

The total supply of the governance tokens is approximately allocated as follows [14]:

- 20% to the core team
- 10% to early investors, advisors and VC's
- 30% to Ithil's treasury (see 6.5),
- 40% to the market, starting from the backing contract (see 6.3)

6.2 IRL adjustments

The passage from Ithil's core protocol and its tokenomics is done through the IRL. As seen in 3.3.4, a given vault's IRL can be adjusted by the governance. If the current ratio is lower than the optimal one, the governance can decide to use part of the treasury's funds to increase it up to the optimal ratio. At the same time, if it is larger, the governance can decrease it, still until the optimal ratio is restored.

In the second case, the funds obtained are *algorithmically* transferred to the backing contract (see 6.3), thus increasing the backed price [15]. In particular, no governance decision is involved in this process: the treasury has no way to withdraw such funds. The effect of this is beneficial to all ITHIL holders, and indirectly to the treasury, which is the primary holder at inception: the treasury is entitled to the funds obtained only in the sense that it is a ITHIL holder, just like all the others.

6.3 Backing

ITHIL is the native token of one or more asymmetric AMM pools whose numeraire are stablecoins. Although a precise treatment of this system would go far beyond the scope of this whitepaper, we can mention here that the **bid price** (or **backed price**) of such pool is calculated as follows. Letting B be the balance of the pool in stablecoins, S the balance of the pool in ITHIL coins, and S_T the total ITHIL supply, then

$$P_{\text{backed}} = \frac{B}{S_T - S}. \tag{13}$$

In the backing pool, any quantity of ITHIL can be exchanged to get the backing stablecoins at the backed price. Conversely, the pool also has a **ask price** $P_{\text{ask}} \geq P_{\text{bid}}$ at which ITHIL tokens can be bought.

It can be shown that with this system, all circulating ITHIL can potentially be redeemed on the backing pool at the backed price, redemptions do not change the backed price, and purchases of ITHIL tokens increase the backed price. In particular, *mathematically, there exists no event which can decrease the backed price.* It can also be shown that any market price outside the pool (for instance,

[14] The precise allocation will be displayed in a dedicated website

[15] Eventually, a swap may be necessary to convert them into the stablecoins backing ITHIL

how much ITHIL is exchanged on an external dex) must lie between P_{backed} and P_{ask}. In particular, since it must be higher than P_{backed}, which is constantly increasing in time, this result represents a guarantee for ITHIL holders that their tokens' value cannot go below such floor.

When IRL adjustments occur (see 6.2), the funds obtained are swapped for stablecoins or ITHIL in the market (depending on which of the two systems brings a greater backed price increase) and injected in the backing pool: in this way, the backed price P_{backed} increases, at the benefit of all ITHIL holders.

The difference function between the bid and ask price is called a **spread model** for the backing pool.

6.4 Boosting

The backing contract is a *booster* (see Section 3.6): the idle funds of the backing are used to boost Ithil's stablecoin APY's. Since, as discussed in 3.6, boosters are the most senior LPs, the backing contract's liquidity has the highest level of protection, thus making the backed price as solid as possible. Since boosting means more APY, which in turn means more liquidity, more usage, more fees and eventually more backing, this system builds a positive feedback mechanism which is beneficial to ITHIL's backed price and ultimately to ITHIL holders.

6.5 Ithil's treasury

Ithil's treasury is made exclusively by ITHIL tokens. This treasury is used to feed the BL of a given Vault, to pay for community mining initiatives and airdrops, and to participate in eventual external investments, managed by Ithil's governance in the form of DAO. At inception, the Treasury will also have the keys to call the `onlyOwner` functions of the contract: the DAO can decide to remove this privilege by a governance vote.