Ithil

A generalised leveraged investment strategies protocol

V1.0.0 - March 11, 2022

Abstract

Ithil aims at introducing undercollateralised leveraged strategies in DeFi - a game changer for traders, liquidity providers and other protocols who can now rely on a variety of investment products to address their needs. Modular and upgradable at its core, Ithil offers users and other protocols leveraged interactions with the DeFi space, enabling an entirely new range of invesment opportunities, acting as an open box financial instrument open to everyone. Liquidity is taken from liquidity providers, who can stake any whitelisted ERC20 token and get a high APY on that same token, and is protected by an efficient and onchain system of liquidations. An innovative backing system mathematically ensures the governance token to always increase in its intrinsic value, thus making Ithil's community stronger as time goes.

Contents

1 Introduction

In the current DeFi world, a large variety of investment opportunities is available, with a new one being presented almost every day. Any user has therefore a wide landscape to choose from, depending on his or her risk appetite and personal ideas. The typical parameters investors look for are the APY (such as in staking-like strategies), the market exposure (such as by holding volatile tokens), the protocol's reliability and security, the liquidity available, personal taste and many other factors difficult to predict or model.

1.1 What is Ithil

Ithil is a protocol which makes the liquidity provider's tokens available to the users, who can then invest much more than their initial capital (that is, with a *leverage*) on a wide choice of DeFi protocols. In this way, Ithil can be considered as a *trait-d'union* between the many protocols, traders and liquidity providers.

1.2 What can be done on Ithil

With Ithil, anyone can

1. Become **liquidity provider** (LP) by staking their tokens and get a solid APY. A big 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 interesting staking opportunity, also for holders of volatile tokens.

2. Become **investor** by placing some collateral in Ithil and then use the LPs' liquidity to perform leverage investments. Ithil allows for high leverages through an internal system of undercollateralized loans: by placing only 100 DAI worth of collateral, a user can perform an investment worth 1000 DAI or more.

3. Become **liquidator** by constantly checking the losing positions and liquidate them in a fully onchain way, thus getting rich rewards.

4. Become part of the **community** by buying and holding Ithil's governance token, or using it in the liquidation process or participating to 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 DeFi landscapes. Some of the differences are the following:

- **Modularity** allows Ithil to list or de-list virtually any strategy with a governance vote. This allows Ithil to be always up to date with the most recent DeFi protocols, offering always the best strategies to the users.

- **High leverage** overcollateralization has always been an important aspect in DeFi loans, which reduces the possibility for users to leverage their funds. [1] Thanks to a system in which the assets never leave the protocol, Ithil makes *undercollateralized* loans possible, thus allowing users to get high leverages and high gains.
- **Possibility of staking any token**: few protocols offer an interesting APY on volatile tokens, and allow only to stake stablecoins or Ethers. Ithil's vaults are instead token-agnostic: any whitelisted token can be staked and it generates APY in the same token.
- **Sustainable high APY** even on stablecoins' vaults, thanks to a sustainable treasury management and a system of internal compensations: when the APY decreases, the governance *algorithmically* increases it, while when the APY increases, the governance is re-charged by decreasing the APY, again algorithmically (no governance vote is ever involved in this process).
- **High capital protection** for liquidity providers: liquidators, the treasury and an algorithmically maintained insurance reserve protect the liquidity, thus assuring long lasting earnings to LPs.
- **Solid tokenomics** which mathematically assures Ithil's governance token value to always increase in time [2] thus rewarding holders committing to the protocol for a long time.

1.4 Ithil's mission

Commitment to offering a sustainable way to interact with the DeFi investment opportunities and get the best out of them is the main focus of Ithil's core team. Ithil presents itself as a benchmark for DeFi strategies, where expert strategists implement easy and secure ways to connect to external protocols and get the most out of them.

In this way, instead of wandering around the huge and complex DeFi world, users can land on Ithil and safely invest on their favourite strategy. Through leverage, Ithil wants to mitigate the intrinsic advantage of wealthy individuals, allowing small users to multiply their exposure.

Building trust is of primary importance, especially in a DeFi world where scammers and pyramid schemes are unfortunately too much popular. This is why Ithil commits to give real value to the protocol: every surplus earned by the protocol will be reinvested internally in the protocol or injected into the backing contract, thus increasing the value of its token and the power of its treasury.

2 Core concepts

In this section we summarize the main concepts of Ithil's workflow, which will be treated more in depth in the following sections.

[1] Or makes this process very costly, particularly through the so-called *loops*

[2] More precisely, to never go below an always increasing floor: see 6.3 for details

2.1 The Vault

The **Vault** is a smart contract which collects the liquidity to be used to perform the leveraged investments required by the users. The liquidity contained in the Vault is made of **whitelisted tokens**, 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 the vault is considered an LP. Such liquidity can be lent to the Strategies, it collects fees, can be redeemed any time [3] 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 Strategies, it collects fees, it cannot be redeemed by anyone [4] and is not protected: it is used to protect the LPL in case a Bad Liquidation Event (see Section 2.6) occurs.

3. **Treasury Owned Liquidity** (TOL). This is injected into the Vault by the governance to bootstrap the protocol and increase LPL's APY. It can be lent to the Strategies, it does not collect fees (they are delivered to the LPL), is protected by IRL and liquidators, and it can be redeemed only by the Treasury after a governance decision.

The liquidity is used by lending it to the Strategies (see 2.2) in the form of undercollateralized loans (see 2.3). Only Strategies can take loans from the Vault and repay them: the `borrow` and `repay` functions are not callable by external addresses. This insures the assets never leave the protocol, and makes undercollateralized loans possible.

Besides managing staking and unstaking LP's tokens, the Vault is also responsible of registering onchain the total loans, of whitelisting tokens and Strategies, to compute the interest rate (see 2.4) of each loan, and to calculate the payoff to give to the user who performed the particular investment.

2.2 Strategies

The **Strategies** are the implementation of the various actions to take to perform a particular DeFi investment. These actions include taking a margin from a user, a loan from the Vault, and calling an external DeFi protocol to obtain some assets.

Ithil implements a modular structure for the Strategies, in which a **Base Strategy** contains all the necessary methods to enforce loan taking and repaying, while the **Strategy Implementation** implements the interactions with the relevant external DeFi protocols. When the external protocol is called, the Strategy swaps Ithil's liquidity, eventually together with the user's margin (see 2.3) to obtain some **assets**, which are then locked into the Strategy and become **held tokens**. The original tokens taken from Ithil to obtain the desired assets

[3]Assuming it is not lent in that particular moment, although it collects high fees if it is. See Section 3.4 for more details.

[4]But if it is too high, the governance can withdraw it to rebalance it. See Section 6.4 for more details.

are **owed tokens**, in that the Strategy *owes* them to Ithil's Vault in order to repay the loan.

The strategy's workflow is summarized as follows:

- The user transfers some tokens as **margin** to the desired Strategy, and chooses the amount of liquidity to be taken from the Vault to increase the investment amount.
- If the amount is not excessive (see 2.4), the Strategy takes the necessary liquidity from the Vault.
- The Strategy calls the relevant external DeFi 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 Strategy. Without this system, it would be impossible to grant an undercollateralized loan (see 2.3).
- The position is part of the Strategy's state, and is visible on-chain. The `quote` public function of the Strategy allows to check the value of the held tokens with respect to the owed tokens: if a position's value decreases below a certain **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 Strategy will call the relevant external DeFi protocols to exchange its held tokens into owed tokens, which are then transferred 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 rather abstract, and can be applied to a wide variety of Strategies. Each time a new strategy is implemented, it must be **whitelisted** by the governance to be able to take loans from the Vault. Only whitelisted addresses can call the borrow and repay functions of the Vault, and ensure the loan is repaid at the end of the investment.

2.3 Undercollateralized loans

One of the most important part of the Strategies' is the process of borrowing liquidity from the Vault and using it to perform investments. Since the liquidity taken may be higher than the margin posted by the user, this has the same effect of a **loan** issued to implement some investment strategy with the Vault's liquidity.

An important takeaway is the fact that *this is not a loan to the user*: Ithil does not transfer any liquidity to the user's wallet until the position is fully closed and the loan repaid. It is instead a loan "from Ithil to Ithil", though the profits are transferred to the user.

In order to perform an investment, the user has to transfer some **margin** to the Strategy. This margin covers the position from eventual losses caused by unfavorable market movements. The margin, just like the assets coming from a given investment, is locked in the Strategy 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's liquidity to perform an investment.

Once the position is opened and the assets are inside the Strategy, the part of assets coming from the user's margin is called the **collateral** placed for the loan. The ratio between the collateral [5] and the loan taken is called the **collateralization** of the loan: in the limit of zero liquidity borrowed from the Vault (no leverage), the collateralization is infinite, and it decreases as the loan increases with respect to the margin amount. The loan is called **overcollateralized** if its collateralization is higher than 1: the loan taken is lower than the margin placed, so the position is virtually riskless. [6] If the collateralization is lower than 1, we say the loan is **undercollateralized**.

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

2.4 Interest rate

When some liquidity is borrowed by the Strategy, the Vault registers the loan in an onchain 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 important 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 Strategy.

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 investment (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 a maximum leverage or a maximum usage, all the risk data are collected together, thus leading to dynamic limits for all such data. In particular, the **maximum leverage** allowed depends on the particular Strategy (risk factor) used, on the usage, and on the insurance reserve available in that moment.

The dynamic interest rate also allows the Vault to automatically 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

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

[6] 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.

3.4), 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.

When fees are generated, an algorithmically computed part is given to the insurance reserve. 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 which 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 different than the optimal ratio, the governance can choose to rebalance it by depositing or 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, or deposit on a ratio larger 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 loss 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 onchain and in a totally decentralized way: anyone can liquidate a given position, as far as the position is **liquidable**, i.e. has a value which is too close to the original loan to be kept open.

The typical liquidation process is summarized 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 lliquidation methods (see 5.3): if Ithil's onchain check confirms the liquidability of the position, the position is closed.
- The proceeds of the closure are transferred to the Vault to repay the loan and relative fees, while the remaining part is split between the user, and the liquidator as a compensation for having liquidated the position (see 5.5 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 (nothing goes to the user in this case).

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 5.5)

2.7 Tokenomics

Ithil's **governance token** ITHIL has the primary function of helping the protocol advance and offer the best services to its users. ITHIL token holders are rewarded in many ways, depending on the usage they do of the token. Moreover, a simple and innovative backing system ensures its value to never fall below an always 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 backing price. This protects the holders from market fluctuations and indirectly rewards holders committing to the protocol for a long time.

3 Liquidity provisioning

As mentioned in 2.1, Ithil's liquidity is obtained from liquidity providers, which 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 of 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 Treasury Owned

Liquidity or TOL. Except for Ithil's treasury, 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 a LP has staked 1000 DAI on the Vault, and afterwards a position which 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 aforementioned LP.

Example 2. If a 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.

LP are essential to Ithil, since without their liquidity, leveraged investments would not be possible. 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 TOL is higher, thus a small usage is compensated by a high Treasury's APY, which attracts new LPs (see 3.5 for more details).
2. If a large amount of LPL is in the Vault, the cost of making riskier investment is lower (see 3.3), thus a lower TOL 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, TOL-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 a 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.

$$LPL_{\text{TKN}} + IRL_{\text{TKN}} + TOL_{\text{TKN}} = B_{\text{TKN}} + L_{\text{TKN}} \tag{2}$$

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). Since the balance is obtainable by the ERC20 method `balanceOf`, we see that the amount of data to be registered onchain is three, with the fourth 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), thus we have (we drop subscripts for notational convenience)

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

In Section 3.4, 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):

$$r = r_{\text{base}} + \frac{L + \max(L - IRL; 0)}{LPL + TOL - IRL} \frac{\beta}{\kappa}. \tag{4}$$

As in Section 3.2, L are the total loans already taken, and LPL, TOL and IRL are as in Section 2.1. 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).

- The **risk factor** β, which is the riskiness of the chosen investment. This has a key importance in liquidations (see 5.2), and has both a governance an an algorithmic component.
- The **base rate** r_{base}, which is the minimal interest rate to be applied (in the particular case of zero loan taken, i.e. $\kappa = \infty$, this makes the interest rate nonzero). It is chosen by the governance.

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* one given LP has staked contribute to that LP gain, and that, given that withdrawals do not change the share price (see 3.2), once the fees are generated, they can be redeemed at any time by that given LP: 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 so 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, following the assumption 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 Redemptions

The wrapped tokens `iTKN` can be redeemed at any time to obtain `TKN`. The amount obtained is calculating 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 term 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 [7] 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 in the other, depending whether the market price is higher or lower than the share price).

[7]We are thinking of external dexes like Uniswap

3.4.1 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` at the same time, the Vault may not have enough liquidity to immediately redeem them.

In this case, the first to redeem will get their `TKN` immediately, but they will make the denominator of Equation (4) 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 to take 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 accumulate so many fees, that it is very profitable to keep them staked.

3.5 Treasury liquidity and APY boosts

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

The governance can, in any time, inject or withdraw some TOL in a given Vault (i.e. in a given token). This liquidity can then be borrowed by strategies, and therefore generate fees, however *these fees are distributed to the LPL and IRL*: in other words, the TOL does not accumulate fees.

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

The TOL 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 TOL 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 TOL, 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`. [8]

The enormous APY obtained as described in Example 3 will then benefit early stakers, who will compete to have the higher portion of the TOL's fees. Notice, however, that *the Treasury's liquidity is not given out to LPs*: except for Bad Liquidation Events (see 2.6), if the Treasury puts some amount M as TOL, then the same amount M can be withdrawn by the Treasury itself at any time, via a governance vote. In this way, the system of APY boosts when the LPL is low can be reiterated in time, rather than being a one-time incentive available shortly after the protocol's launch: Ithil's sustainability is assured.

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

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. [9] 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, which 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 being delisted by the governance, following market conditions and sentiments. The architecture in place allows for an 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 which 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 either in `TKA` or in `TKB`.

2. The user chooses a quantity A an 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.

4. The Strategy computes the **collateral** placed by the user for the loan. If the margin is in `TKA`, then the collateral is simply 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`.

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

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 onchain 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 undercollateralized loan from the Vault in a secure way (the user cannot run away with the assets).

4.1.2 Closing a position

The address of the user opening a position 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 [10] 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.

In a profitable investment with a `TKA` margin, the amount of `TKA` obtained at the closure is higher than the ones 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 which is much higher than the margin posted, thanks to the funds borrowed from the Vault.

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

4.1.3 Editing a position

While a position is open, its owner can decide in 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.* [11] 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 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 all loans are correctly repaid and that the due fees are generated, so that this core feature is respected regardless on 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 interest rate of the loan, 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 [12] which have the final effect of having an *exposure* to some external market movement. Such movement can be, in principle of any type: from 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 an exposure to real-world assets like in synthetics-related strategies, to raffles, etc...

Ithil's modularity allows to list or de-list Strategies at any time, through a governance decision. The fact that much of the logic is contained in the

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

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

Base Strategy significantly reduces the effort necessary to code a particular implementation.

The Base Strategy has three `virtual` functions, `_openPosition`, `_closePosition` and `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`. [13] 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 calling a function in the Base Strategy contract.

4.4 Example of strategies

In what follows, two examples of strategies are given. It is important to understand that these two are just examples, and they are inserted here *for illustration 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 rather different in nature, riskiness, and implementation, they illustrate well the flexibility of Ithil's architecture.

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

4.4.1 Leveraged yield farming

4.4.2 Margin trading

5 Liquidations

5.1 Liquidators

5.2 Risk factor and liquidation score

5.3 Types of liquidations

5.4 Liquidator's rewards

5.5 Staking governance tokens

6 Tokenomics

6.1 The governance token

6.2 IRL adjustments

6.3 Backing and perpetual auction

6.4 Ithil's treasury

6.5 Airdrops and community mining

6.6 Ithil's DAO

7 Glossary