Hoyu Protocol

Abstract

Since the advent of decentralized finance (DeFi), two applications have found product-market fit on every blockchain which supports DeFi. These are automated market-makers (AMMs) and money markets, often called lending protocols.

Innovation in AMMs has led to many variants of the original model based on Uniswap's $xy = k$ constant product formula; innovation in lending markets has been slower. Despite their popularity, lending protocols still have one major limitation: they are highly selective in their listings, excluding almost all crypto assets.

This is for two reasons:

1. Lending protocols reference external price feeds which are not secure enough to depend on.

2. Due to their architecture, existing lending protocols are unable to check assets' liquidity directly; most rely on whitelisting, only listing sufficiently liquid assets in an attempt to prevent *bad debt*.

Hoyu solves both of these problems by combining DeFi's two major applications; Hoyu references the price and liquidity in its own AMM pools to dynamically determine lending limits. Due to Hoyu's architecture, any token with sufficient liquidity in a Hoyu AMM pool can be used to borrow stablecoins in the Hoyu protocol without risk of default to lenders or the protocol itself.

1 Introduction

The main limitation of existing lending protocols is the number of assets which can be loaned and borrowed.

A peer-to-peer architecture provides a partial solution. However, they still suffer from the inefficiencies common to over-the-counter markets, counterparty discovery is difficult, even where liquidity exists. Furthermore, peer-to-peer lending protocols force lenders to take on all of the volatility risk of borrowed and loaned assets.

Other lending protocols limit the number of assets through curation; a small number of assets deemed liquid enough to justify *mark-to-market* accounting are allowed to trade on the lending market.

Multiple variations on this curation process have been tried, yet still fail to prevent bad debt[1], where a borrower takes out a loan which is not profitable to repay.

2 Problem

The creation of bad debt in lending protocols stems from two dependencies: liquidity assumptions and external price oracles.

Before addressing the protocol's solution to the problem of bad debt, we must first understand its root causes.

2.1 Liquidity assumptions

When approving a particular token for use as collateral in a lending protocol, decision-makers will often look at factors like volume, number of holders, and volatility to get an idea of market risks[2].

The goal of such an assessment is to provide a measure of confidence that there is sufficient liquidity in the band between the spot price of an asset and its liquidation threshold price such that it may be *marked to market.*

While these metrics may be directionally correct, they are merely time-weighted proxies, rather than direct measurements of liquidity.

2.2 Price oracles

Dependence on external price oracles is another vulnerability underlying bad debt in lending protocols. Oracle manipulation attacks arise from the use of external oracles[3].

Oracle manipulation attacks change the price data given by an oracle either by targeting price feeds which an oracle references, or by replacing an oracle, or bribing the oracle directly in order to manipulate prices given to the lending market.

Both of these categories of vulnerabilities are avoided in Hoyu's design.

3 Hoyu AMM pools

Hoyu AMM pools follow the $xy = k$ constant product AMM formula, with a few key differences, as we make clear in the following sections[4].

3.1 Token designations

Like Uniswap pools, Hoyu pools are created permissionlessly, and only one pool for a given pair can exist. However, tokens in Hoyu pairs have distinct roles. Hoyu token pairs are canonically designated ALT/CUR - abbreviations for altcoin and currency, respectively.

By convention, ALT can be any ERC-20, and tokens designated CUR are stablecoins. Only the CUR token in a pair can be borrowed, while the ALT is used as collateral.

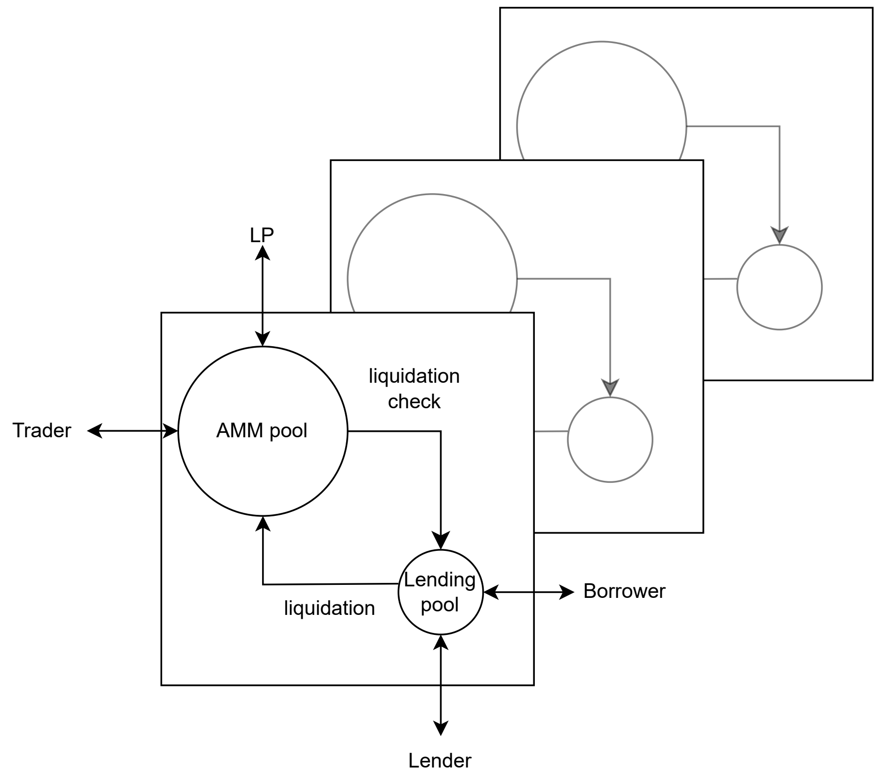

Figure 1: Each Hoyu ALT/CUR pair has 2 modules: an AMM pool, and a lending pool.

Technically, any ERC-20 can be either token in the pair, but these token designations serve an important design function: to prevent the shorting of ALT on Hoyu lending markets.

To preserve Hoyu's credible neutrality and permissionlessness nature, these conventions are kept by the Hoyu front end, rather than enforced at the smart contract level.

3.2 Swaps

Swaps check for liquidations before executing in order to keep the lending market updated on the state of reserves in the AMM pools.

This design enables the Hoyu lending market to use the spot price from Hoyu pools to calculate borrow amounts and enforce liquidations without the possibility of incurring bad debt [Appendix A].

3.3 Fees

Traders pay 0.3% of the swapped amount to the pool, which goes to that pool's liquidity providers.

3.4 Liquidity providing

The liquidity provider (LP) experience in Hoyu pools is slightly different than in Uniswap. Providing tokens is similar, but burning LP tokens has different mechanics.

In order to prevent *liquidation attacks*, tokens withdrawn from a Hoyu LP position are subject to a 7-day linear unlock period [Appendix B]. An LP is able to redeem unlocked tokens early with partial redemption amounts calculated at the time of withdrawal.

4 Lending market

Loans are taken out against CUR deposited by lenders into a lending pool. Deposited CUR accrues interest proportional to pool utilization.

Depositors can borrow any CUR that has sufficient reserves in the AMM pool and unutilized CUR in the lending pool, for the corresponding pair.

To avoid creating bad debt, the amount of CUR that can be borrowed from a lending pool is limited [Appendix A]. For a given ALT/CUR pair, a maximum of 10% of the CUR's liquidity in the AMM pool can be borrowed from the corresponding lending pool against the ALT, deposited to the lending market.

4.1 Lending

Any CUR token can be loaned out via lending pools. Lenders deposit CUR tokens to be borrowed against a particular ALT as collateral.

Pools are isolated from each other. Like Hoyu AMM pools, there is only one lending pool for each ALT/CUR pair. Likewise, a user's borrowing positions are also isolated from one another, meaning that the health of a borrowing position in one pool doesn't affect the health of another.

On depositing CUR tokens into a lending pool, an ERC-4626 vault, lenders receive hTokens (e.g. hUSDT) which give holders a *pro rata* share of CUR tokens in the associated pool[5].

4.2 Borrowing

A borrower may deposit ALT collateral to the Hoyu lending market and can borrow *up to* 1 - the maximum LTV in the CUR token for the ALT/CUR market, providing there is enough available CUR in the ALT/CUR lending pool [Appendix C].

This maximum borrowable amount, λ (see System Parameters in Appendices) is the same for all Hoyu pools. Only CUR tokens can be borrowed; conversely, only ALT tokens are used as collateral.

4.3 Interest

Interest accrues to a borrower's CUR position at 15% annually. For lenders, interest on loaned CUR is directly proportional to the utilization of lending pool funds, e.g. 50% utilization accrues 7.5% interest to a lender's deposited tokens.

Interest may be claimed at any time, as long as there is CUR available in the corresponding vault. Interest accumulates with every second, and even when a loan is not yet repaid, accrued interest can already be claimed, as the accrued value is immediately included when accounting for total assets in the ERC-4626 vault.

4.4 Liquidations

Loans are liquidated automatically, and occur in two cases:

1) On price drops of ALT/CUR 2) On decreases in liquidity such that a 10.2% threshold is crossed

On liquidation of a loan, the ALT collateral is swapped through the AMM pool for CUR. The amount of ALT to be liquidated is calculated in a liquidation such that only the ALT necessary to cover the borrowing fees, plus interest is actually liquidated. After a liquidation, the remaining ALT collateral in a position is returned to the borrower. The CUR from the liquidation is availed to Lenders in the Lending pool for redemption.

Borrowing positions are such that liquidations trigger other liquidations, but Hoyu is designed in a way that makes these cascades finite [Appendix F].

4.5 Virtual reserves

In case of liquidations, the liquidated collateral ALT is swapped for CUR in the ALT/CUR pool. In order to prevent liquidation attacks, *virtual reserves* are introduced.

Virtual reserves are used in calculations shortly after a liquidation takes place [Appendix D]. For purchases of ALT (but not CUR) after a liquidation, calculations take place *as if* the liquidated collateral were not traded into the pool. Instead, liquidated tokens are unlocked into a pool linearly over a 1-hour period.

Virtual reserves are instantiated to prevent holders of a token from performing a particular type of attack, selling ALT to liquidate borrowers and instantly buying it back for a risk-free trade [Appendix E].

5 Protocol fees

A fee is taken each time a borrow position is initiated. This fee, taken in CUR is the greater of 0.5% of the borrow amount, or 5 CUR tokens.

On chains where a Hoyu utility token (HOYU) exists, the CUR fee is sent directly to the HOYU/CUR pool, rewarding the LPs of that pool while also effectively increasing HOYU price.

On chains where a Hoyu token does not exist, this fee is sent to the corresponding lending pool, able to be claimed by hCUR token holders on redemption, with their principal and accrued interest.

6 Conclusion

Hoyu's unique protocol design solves the challenges common to lending markets; Hoyu users can borrow stablecoins against any asset at a high LTV, with zero risk of bad debt to lenders or the protocol itself.

References

1. Bad debt on lending markets https://bad-debt.riskdao.org/
2. Aave asset risk methodology https://docs.aave.com/risk/asset-risk/methodology
3. Oracle manipulation attacks https://github.com/0xcacti/awesome-oracle-manipulation
4. Uniswap v2 repository https://github.com/Uniswap/v2-core/blob/master/contracts/UniswapV2Pair.sol
5. ERC-4626 token standard https://eips.ethereum.org/EIPS/eip-4626

Appendices

These appendices provide technical justification that the properties in the main body of the whitepaper are satisfied. They contain both formal math proofs and high-level explanations.

We will utilize the following notation throughout the appendices.

System Variables

Symbol	Meaning
R_{ALT}	Reserves of ALT in DEX
R_{CUR}	Reserves of CUR in DEX
T_{ALT}	Amount of ALT involved in a trade on the DEX
T_{CUR}	Amount of CUR involved in a trade on the DEX
L_{ALT}	Total amount of ALT provided for all loans in the Lending Market
L_{CUR}	Total amount of CUR required to pay back all loans on the Lending Market
L^k_{CUR}	Amount of CUR required to pay back specific Loan k.
L^k_{ALT}	Amount of ALT deposited as collateral for specific Loan k.

System Parameters

Hoyu has several parameters. In general, these parameters can take arbitrary values. In this paper, the parameter values are those chosen for the initial version of the Hoyu Protocol, Hoyu v1. These values are recorded below.

Symbol	Meaning	Hoyu v1 Value
ν	Maximum Loan-to-Value	0.894
μ	Individual Loan Limit	0.10
λ	Global Loan Limit	0.102
d_{max}	Maximum Drop in Spot Price from Loan Liquidations (determined by λ)	0.195
L_{fee}	Protocol fee for loan origination	Maximum of 0.5% of loan amount and 5 CUR
T	Virtual Reserves Decay Time Interval	1 hour
ϕ	DEX Trading Fee	0.003
B_D	Length of Total LP Token Burn Duration	1 week
B_L	Length of LP Token Burn Interval	12 hours
B_N	Number of LP Token Burn Intervals	14 intervals
w	Minimum tick width	1.00271

Important Inequalities

- **Maximum Loan-to-Value.** The maximum possible Loan-to-Value ratio for a single loan, denoted ν, says that relationship between L^k_{CUR} and the collateral amount L^k_{ALT} must satisfy the inequality

$$L^k_{\text{CUR}} \leq \nu \frac{R_{\text{CUR}}}{R_{\text{ALT}}} \cdot L^k_{\text{ALT}}. \tag{1}$$

- **Individual Loan Limit.** When a single loan L^k is created, the amount of the loan in CUR must not exceed the Individual Loan Limit μ, i.e.

$$\frac{L^k_{\text{CUR}}}{R_{\text{CUR}}} \leq \mu. \tag{2}$$

- **Global Loan Limit.** The maximum possible ratio of loaned CUR (including both principal and interest) for all loans, denoted λ, is set at $\lambda = 0.102$. In other words,

$$\frac{L_{\text{CUR}}}{R_{\text{CUR}}} \leq \lambda. \tag{3}$$

Appendix A: System Properties and Guarantees

In this section we demonstrate guarantees that certain system properties will always be maintained. In particular,

- Lenders cannot lose CUR; any loan liquidation has sufficient collateral to recover the CUR related to that loan.
- Borrowers have no incentive to create loans they do not plan to repay.

The Protocol Avoids Bad Debt

Bad debt occurs when a loan liquidation results in loss of tokens provided by a lender to a market. In this case, lenders could be put in a position where they are not able to reclaim the full value of their original deposit.

Example of Bad Debt

Suppose the Protocol has a maximum Loan-to-Value of $\nu = 0.894$. the DEX contains 1,000,000 CUR And 1,000,000 ALT, for a spot price of 1.0 CUR/ALT. A borrower takes out a loan for 150,000 CUR with collateral of 168,000 ALT. The LTV value of this loan $\frac{150,000}{168,000} \approx 0.893$ is below the required $\nu \approx 0.894$ threshold, so a loan can be created. If the loan is liquidated shortly thereafter, then per $xy = k$, the Lending Market will receive

$$T_{\text{CUR}} = \frac{0.997 \cdot (1,000,000) \cdot (168,000)}{1,000,000 + (0.997)168,000} \tag{4}$$

$$\approx 143,466 \text{ CUR.} \tag{5}$$

Such a liquidation would create bad debt, as the Lending Market receives less than the 150,000 CUR it was owed on the loan. There is nothing unique about the values in this example – a similar calculation will show that it is always true that when a loan is close to the maximum LTV value, allowing too much (in this case, 15%) of the DEX CUR to be loaned out will result in bad debt if the loan is liquidated.

How Hoyu Prevents Bad Debt

The Hoyu Protocol prevents the creation of bad debt by limiting both LTV for an individual loan, and the amount of CUR that can be loaned at one time across the entire lending market.

In this section, we present a mathematical argument that the two system parameters ν (maximum LTV) and λ (maximum collateral loan limit) are sufficient to ensure that no bad debt will be created.

In the worst case, an unliquidated loan will be right at the LTV limit ν, meaning we would have

$$\frac{L_{\text{CUR}}}{\frac{R_{\text{CUR}}}{R_{\text{ALT}}} L_{\text{ALT}}} = \nu \tag{6}$$

or, rewritten another way:

$$L_{\text{ALT}} = \frac{1}{\nu} \frac{L_{\text{CUR}}}{R_{\text{CUR}}} R_{\text{ALT}}. \tag{7}$$

If such loan is liquidated, the Lending Market will receive this L_{ALT} as a trade of ALT for CUR, sending back a trade amount

$$T_{\text{CUR}} = \frac{(1-\phi) L_{\text{ALT}} R_{\text{CUR}}}{R_{\text{ALT}} + (1-\phi) L_{\text{ALT}}}. \tag{8}$$

Our goal is to show that

$$T_{\text{CUR}} \geq L_{\text{CUR}}. \tag{9}$$

This means that the amount of T_{CUR} will always be at least as large as L_{CUR}, i.e. the liquidated loan's CUR amount will be covered by the T_{CUR} received in exchange for the liquidated collateral L_{ALT}.

Based on our worst-case-collateralization assumption, we can replace L_{ALT} with $\frac{1}{\nu}\frac{L_{\text{CUR}}}{R_{\text{CUR}}}R_{\text{ALT}}$. This results in

$$T_{\text{CUR}} = \frac{(1-\phi)\frac{1}{\nu}\frac{L_{\text{CUR}}}{R_{\text{CUR}}}R_{\text{ALT}}R_{\text{CUR}}}{R_{\text{ALT}} + (1-\phi)\frac{1}{\nu}\frac{L_{\text{CUR}}}{R_{\text{CUR}}}R_{\text{ALT}}}. \tag{10}$$

Since we know that $\frac{L_{\text{CUR}}}{R_{\text{CUR}}} \leq \lambda$, replacing $\frac{L_{\text{CUR}}}{R_{\text{CUR}}}$ with λ in Equation 10 corresponds to the worst-case scenario when the loaned collateral is at the maximum allowed amount. This leads to the inequality

$$T_{\text{CUR}} \geq \frac{(1-\phi)\frac{1}{\nu}\frac{L_{\text{CUR}}}{R_{\text{CUR}}}R_{\text{ALT}}R_{\text{CUR}}}{R_{\text{ALT}} + (1-\phi)\frac{1}{\nu}\lambda R_{\text{ALT}}}. \tag{11}$$

Now we can do a bit of algebraic simplification in the numerator on the preceding inequality, leading to factorization in the denominator, as follows:

$$\begin{aligned}
T_{\text{CUR}} &\geq \frac{\frac{(1-\phi)}{\nu}\frac{L_{\text{CUR}}}{\cancel{R_{\text{CUR}}}}\cancel{R_{\text{CUR}}}R_{\text{ALT}}}{R_{\text{ALT}} + \frac{(1-\phi)\lambda}{\nu}R_{\text{ALT}}} && (12)\\
&= \frac{\left(\frac{(1-\phi)}{\nu}L_{\text{CUR}}\right)\cancel{R_{\text{ALT}}}}{\left(1+\frac{\lambda(1-\phi)}{\nu}\right)\cancel{R_{\text{ALT}}}} && (13)\\
&= \left(\frac{\frac{(1-\phi)}{\nu}}{1+\lambda\left(\frac{(1-\phi)}{\nu}\right)}\right)L_{\text{CUR}}. && (14)
\end{aligned}$$

It is important to that the expression being multiplied by L_{CUR} is constant, and does not depend on the state of the pool, the size of the loan, or any other system variable quantity. This multiplier depends only on the values of the constant system parameters ϕ, ν, and λ. If these parameters are chosen to ensure that

$$\left(\frac{\frac{(1-\phi)}{\nu}}{1+\lambda\left(\frac{(1-\phi)}{\nu}\right)}\right) \geq 1, \tag{15}$$

then we will be guaranteed that it is always true that

$$T_{\text{CUR}} \geq L_{\text{CUR}}. \tag{16}$$

The values for these parameters are chosen by the Hoyu Protocol so that $T_{\text{CUR}} \geq L_{\text{CUR}}$ always holds, which means there will always be enough CUR received from the trade of the collateral ALT to cover the entire amount of CUR owed on the loan. It is impossible to create bad debt.

It is important to note, however, that in practice, the Protocol will only liquidate the amount of ALT required to cover the loaned CUR, i.e. the amount of CUR sent from the DEX to the Lending Market on a liquidation will always be exactly the same as the amount of CUR owned. In general, there may be some amount of ALT left over after a liquidation, which the borrower can still reclaim.

No Incentive to Abandon a Loan

A User in any lending market might be tempted to try a "backdoor trade" of ALT for CUR by executing the sequence of actions:

- Taking out a loan for CUR, providing ALT as collateral.
- Intentionally abandoning the loan to liquidation, keeping the loaned CUR (and losing the ALT collateral).

It would be bad if there were any incentive for users to do this as a type of trade. Loan liquidations cause the system to do additional work, including the possibility of inducing further liquidations.

Fortunately, the mathematical argument in the preceding subsection carries over here: T_{CUR} gives the amount of CUR that would be received for trading an amount L_{ALT} of ALT, while L_{CUR} corresponds to the amount of CUR obtained by abandoning a loan with L_{ALT} as collateral. Since $T_{\text{CUR}} > L_{\text{CUR}}$, **trading is always the better strategy** in terms of CUR amount obtained.

Another incentive to trade is that there is no limit to how much ALT can be traded with the AMM, while there is a limit on the size of a new loan.

Price Impact of Liquidations

Since liquidations will have impact on DEX spot price just like any other trade, it's useful to record some facts about what those price impacts can look like.

Observation 1. *Suppose a trade results in a constant-product AMM sending a proportion $\Delta p_C \in [0,1]$ of its reserve CUR to a trader for the appropriate amount of incoming ALT. Then the resulting percentage in spot price is*

$$\Delta P = \frac{(1-\Delta p_C)^2}{(1-\phi)} - 1 \tag{17}$$

Determining the value of $d_{\max}$ is important for quantifying the maximum impact that liquidations in the Lending Market can have on the DEX spot price.

Observation 2. *If d_{max} denotes the maximum percentage change in spot price from a liquidation event, then d_{max} is completely determined by the global loan limit λ.*

Proof. The value of $d_{\max}$ comes from plugging λ into the expression for Δp_C given above. □

If the Loan Limit λ is set to $\lambda = 0.102$, then $d_{\max} \approx 0.195$.

Appendix B: Withdrawal of Liquidity Provision Shares

Mechanics

The tokens available on the DEX are provided by *Liquidity Providers* (LPs) in proportions determined by the current spot price. In exchange, the LP receives *LP tokens* which can be claimed later for an appropriate portion of the pool's assets.

When the LP wishes to reclaim their share of the assets in the pool, they initiate a *burn.* Burns in Hoyu are somewhat different than in other AMM protocols. In a typical AMM, LP tokens are burned instantaneously and underlying tokens are withdrawn immediately. In Hoyu, the underlying tokens are gradually unlocked.

On a technical level, the burn process is governed by two parameters set by the protocol:

- the *burn interval length* B_I, a time interval measured in e.g. hours.
- the *burn interval number* B_N, corresponding to the number of burn intervals that the burn should last.

Regardless of when a burn event is initiated, it will always end on a block whose number is a multiple of B_I.

For a burn event beginning at t_{start}, the burn will end at the timestamp

$$t_{\text{end}} = \left(\left\lfloor \frac{t_{\text{start}}}{B_I} \right\rfloor + B_N \right) \cdot B_I. \tag{18}$$

Notice that this guarantees that the end timestamp t_{end} occurs on a multiple of B_I.

If an LP is withdrawing a total amount of A_{total} LP tokens, the amount burned on each burn interval will be calculated proportional to the duration of the burn.

The *burn duration* B_D can be calculated as $t_{\text{end}} - t_{\text{start}}$. If A_{interval} represents the amount burned on each interval, then A_{interval} can be be calculated as:

$$A_{\text{interval}} = \frac{A_{\text{total}}}{B_D}. \tag{19}$$

This formula captures intuitive behavior: as the burn progresses, the corresponding amounts of ALT and CUR become available for the LP to claim (e.g. if 10% of the total burn time has passed, then 10% of the underlying tokens are available to the LP).

If more than one LP token burn event is active at a given time, their burn rates and amounts are added during the duration of their overlap.

Example: Suppose an LP currently owns 100 LP tokens for a liquidity pool that has 1000 total LP tokens, with 200 CUR And 800 ALT. The system has set burn parameters with the burn interval length $B_L = 12$ hours and burn interval number $B_N = 14$.

The LP begins the withdrawal of their 100 LP tokens at timestamp $t_{\text{start}} = 28$, will end at timestamp

$$\begin{aligned} t_{\text{end}} &= \left(\left\lfloor \frac{28}{12} \right\rfloor + 14 \right) \cdot 12 && (20) \\ &= 192 && (21) \\ & && (22) \end{aligned}$$

lasting for a duration of 164 hours.

On each block, there will be a total of

$$A_{\text{block}} = \frac{100}{164} \tag{23}$$

LP tokens that can be burned.

Implications of the Burn Mechanics

There is no way to prohibit "insider trading" or other actions based on information asymmetries in a permissionless system. Thus, one of the major risks to ordinary system participants is that a Liquidity Provider may take sudden action based on accurate knowledge about the ALT token that is not yet available to the rest of the market.

The gradual nature of the burn and redemption limits the ability of LPs to "rug pull" other users. In effect, the potential for information asymmetry is balanced by a corresponding reaction time asymmetry: traders and borrowers can execute their actions instantaneously, aware of the signal provided by the LP's withdrawal actions.

These trade actions has an impact on the value of the LP tokens being withdrawn. In general, it is reasonable to calculate the total value of a liquidity pool in terms of CUR as

$$V_P = p_{\text{ALT}} \cdot R_{\text{ALT}} + R_{\text{CUR}} \tag{24}$$

where p_{ALT} represents the "true price" of ALT. Using the spot price for ALT as the true price gives $p_{\text{ALT}} = \dfrac{R_{\text{CUR}}}{R_{\text{ALT}}}$, which would give a value of $V_P = 2 \cdot R_{\text{CUR}}$. Since each LP token represents a proportional claim on the overall pool value, LPs need to take such factors into account before deciding to withdraw.

If an LP withdraws tokens to profit from an anticipated shock to the price of ALT, traders will still have the opportunity to execute their trades instantaneously. These traders also benefit from the fact that the LP's liquidity is not yet withdrawn. Loan liquidations follow a similar logic. A large LP token withdrawal which leads to many loan liquidations will change the proportion of the pool to have more ALT and less CUR. In this scenario, the value of LP tokens decreases with the price of ALT.

Ultimately, an aware borrower will decide whether it is more advantageous to allow liquidation or repay their loan based on all the information available. Extending the time over which LP tokens are burned decreases the ability for a Liquidity Provider to take actions based on private information.

Appendix C: Virtual Reserves

Suppose a trade occurs which triggers a liquidation event at a *start time* t_{start}, resulting in total changes of ΔR_{ALT} and ΔR_{CUR} to a pool's reserves. This will trigger the *virtual reserves condition* to be in effect for a predefined time interval T, a system parameter set by the protocol.

During this time period, all *buy ALT* events are calculated using quantities V_{ALT} and V_{CUR} instead of the actual reserves R_{CUR} and R_{ALT}, where V_{ALT} and V_{CUR} are defined as:

$$V_{\text{ALT}} = R_{\text{ALT}} + \frac{\Delta R_{\text{ALT}}}{T}(t - t_{\text{start}}) \tag{25}$$

and

$$V_{\text{CUR}} = R_{\text{CUR}} + \frac{\Delta R_{CUR}}{T}(t - t_{\text{start}}). \tag{26}$$

These are linear functions, beginning at the pre-liquidation amounts of R_{CUR} and R_{ALT} at time t_{start}, and ending at the post-liquidation amounts at the end of the time interval T. The effect of this is a gradual change in the ALT price for *buy ALT* events.

If no Virtual Reserves are in effect, a User who wishes to trade an amount T_{CUR} of CUR to buy ALT would receive

$$T_{\text{ALT}} = \frac{(1-\phi)R_{\text{ALT}}T_{\text{CUR}}}{R_{\text{CUR}} + (1-\phi)T_{\text{CUR}}}. \tag{27}$$

However, with Virtual Reserves in effect, they would instead receive

$$T^V_{\text{ALT}} = \frac{(1-\phi)V_{\text{ALT}}T_{\text{CUR}}}{V_{\text{CUR}} + (1-\phi)T_{\text{CUR}}}, \tag{28}$$

which is a smaller amount than T_{ALT} since $R_{\text{ALT}} \leq V_{\text{ALT}}$ and $R_{\text{CUR}} \geq V_{\text{CUR}}$.

The precise quantitative impact depends on the amounts involved in the liquidations. This depends on both the amount of ALT sold to the DEX, as well as the structure of the loans that are liquidated. However, the qualitative impact is both clear and crucial: **the possibility of a risk-free liquidation-causing trade is eliminated.**

Instead, a trader who sells ALT hoping to force a liquidation must now enter a waiting game, during which the actions of other traders also introduce uncertainty regarding the price trajectory of ALT. For instance, if another trader buys ALT during the Virtual Reserves period, it could undercut the original trader's desired price action. Additionaly, other liquidations could further affect Virtual Reserves, adding to the delay time to reach a desired buy price for ALT.

Although it may still be possible in certain circumstances for a trader to profit from liquidations they have caused, the Virtual Reserves mechanism protects the system by making such an action subject to both delay and potential risks.

Appendix D: Triggering Liquidations to Manipulate Price

A potential attack on the Hoyu system consists of making trades with the DEX to force liquidations, hoping to make the DEX conditions more favorable for future trades. The precise actions would be:

1. User initiates a *sell ALT* event with the DEX, which reduces both spot price and the amount of reserve CUR in the DEX.
2. The drop in spot price reduces the value of loans on the Lending Market. This could result in the minimum LTV condition being violated for some loans, triggering their liquidations.
3. The drop in CUR could also result in the Global Loan Limit being breached, resulting in liquidations.
4. These liquidations further increase the supply of ALT in the pair, reducing spot price even more. At this point User initiates a *buy ALT* event, as the price of ALT is now cheaper than their original sell price.

There are two mechanisms in the Hoyu system that discourage such an action:

- Liquidations are processed before a User trades, making the trade less profitable.
- The use of Virtual Reserves, triggered by liquidations, leading to price being impacted gradually rather than instantaneously.

Appendix E: Complexity of Calculating Loan Liquidations

In the Hoyu system, a trade in the DEX can cause liquidations in the Lending Market. However, such a liquidation is a trade in its own right, it is possible for the first liquidation to cause another liquidation, leading to a longer *liquidation cascade* of several liquidation cascades. Each liquidation requires checks of which loans should be liquidated, and these liquidation checks require time and gas for computation.

For any given trade, how many liquidations occur in a cascade will depend on the structure of the existing loans. In practice, there are reasons to expect that the conditions for a "perfect storm" of long liquidation cascades to be rare. However, even in the worst-case scenario, the Hoyu system is designed to remain fully operational.

Organization of Loans

The Hoyu protocol organizes loans in a particular way that helps to ensure efficient operation.

Loans are Organized in Ticks

Loans in the Hoyu system are organized into structures called *ticks*. Each tick corresponds to a range of spot prices in the DEX.

When a loan is created, it is placed into the tick containing the spot price at which the loan's LTV would force its liquidation. Loans in the same tick have similar LTVs, so that all loans in a tick will be liquidated together.

Width and Number of Ticks

The *width* of a tick is the ratio of its top spot price to its bottom spot price. The minimum tick width w is a system parameter set by the protocol. This minimum tick width, together with the bounded range of numerical values allowed by the Ethereum Virtual Machine, lead to a finite number of ticks available for the placement of loans. The precise value of this number is completely determined by the choice of w (and the limits of the EVM).

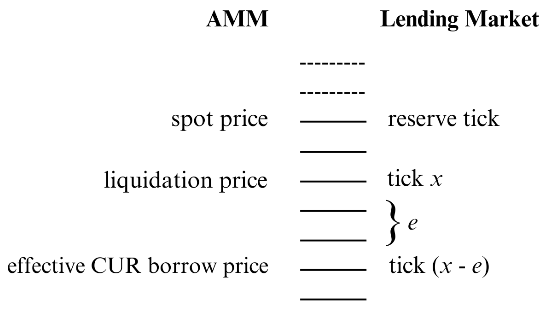

Figure 2: A visual representation of ticks

Under normal circumstances, the reserve tick in the lending market maps to the current spot price in the AMM. Loans may be placed into ticks with liquidation prices anywhere below the reserve tick. The tick into which a loan is placed maps to its liquidation price, with the maximum LTV falling e number of ticks below that one, where e is the number which exponentiates (1 + w, the tick width) to yield (1 - maximum LTV).

Ticks Help Guarantee Computational Efficiency

The tick corresponding to the current DEX spot price is called the *reserve tick*. As trades (whether user-initiated or as the result of liquidations) are executed and change the DEX spot price, the reserve tick will move accordingly.

When a trade is executed, the system checks a sequence of potentially affected ticks below the current reserve tick to determine whether they will be liquidated. In the absolute worst case, the number of liquidation check calculations is bounded by something on the order of the number of total ticks. This is far better than a naive loan-by-loan approach whose computational demands would scale with the number of loans, admitting no *a priori* bound.

Computational Complexity of Liquidation Cascades

The computational picture is helped even more by the fact that, in practice, the number of potentially affected ticks due to a single liquidation is usually much smaller than the total number N_{tick} of ticks in the system.

Recall that there are two reasons a loan can be liquidated:

- An individual loan L^k can undergo an *LTV breach*, where

$$\frac{L^k_{\text{CUR}}}{L^k_{\text{ALT}}} \geq \nu \frac{R_{\text{CUR}}}{R_{\text{ALT}}}.$$

- The entire system can undergo a *global loan limit breach*, where

$$L_{\text{CUR}} \geq \lambda R_{\text{CUR}}.$$

In principle, there are four different kinds of liquidation cascades that can occur:

1. LTV Breach $\rightarrow$ LTV Breach;
2. LTV Breach $\rightarrow$ Loan Limit Breach;
3. Loan Limit Breach $\rightarrow$ LTV Breach;
4. Loan Limit Breach $\rightarrow$ Loan Limit Breach.

The key guarantee we have in terms of system performance is that **the number of liquidation checks needed for each type of cascade has a clear finite bound**. This fact can be seen by considering each possibility carefully.

LTV $\rightarrow$ LTV

Finding the Length of Different Tick Zones

For liquidation cascades in general and LTV $\rightarrow$ LTV cascades in particular, it is helpful to understand the ranges of ticks, and how the system moves through them.

Going down a single tick in price is equivalent to multiplying the current spot price by the inverse of the current tick width. While tick width can vary somewhat, it will always be at least the minimum tick width w.

The overall structure of ticks that may need to be checked for liquidation is desscribed below:

- The current reserve spot price sits in the Reserve Tick.
- Immediately below the Reserve Tick, there is a series of ticks where good loans cannot be, which we call the the *Liquidation Buffer*. Since we are considering LTV breaches, any loans which

satisfy $\frac{L^k_{\text{CUR}}}{L^k_{\text{ALT}}} \geq \lambda \frac{R_{\text{CUR}}}{R_{\text{ALT}}}$ would definitely need to be liquidated if they haven't already. The number of ticks in this range can be found by solving

$$w^{-k} = \nu \tag{29}$$

where ν is the maximum LTV Limit.

- Below the Liquidation Buffer, there's a block of ticks in which the loans could be liquidated. However, as we are primarily interested in LTV→ LTV cascades, there is a natural range to consider here as well. Recall that if all the outstanding loans are liquidated, there is a maximum spot price drop of $d_{\max}$ that depends on the Global Loan Limit λ (established in Observation 2.) Finding how many ticks will be traversed before reaching this range, is equivalent to solving for the smallest of value of b_1 such
$$(w)^{-b_1} = (1 - d_{\max}). \tag{30}$$
- We are guaranteed that this type of liquidation cascade will cover no more than b_1 ticks. It is important to note here that b_1 is a constant value that depends on the values of the constants w $d_{\max}$ (which in turn depends only on the value of the constant λ). As long as λ and w are chosen appropriately, it is possible to guarantee that the number of ticks that need to be checked in each LTV → LTV cascade is reasonable.

As an example, suppose the system is implemented with parameters $\lambda = 0.102$ and $w = 1.00271$. Solving for b_1 here gives the bound $b_1 = 81$ as the maximum number of ticks to be checked for liquidation in an LTV → LTV liquidation cascade.

LTV → Loan limit

By construction, as long as the global loan limit is respected, liquidating a loan improves the health of the system. In other words, such a liquidation always decreases the ratio of loaned CUR to reserve CUR. Let us show this more formally.

Claim 1. *Let U be defined as*

$$U = \frac{L_{CUR}}{R_{CUR}}. \tag{31}$$

Suppose a loan liquidation occurs which removes $\Delta L_{CUR} > 0$ from both L_{CUR} and R_{CUR}. Let

$$U_{new} = \frac{L_{CUR} - \Delta L_{CUR}}{R_{CUR} - \Delta L_{CUR}}. \tag{32}$$

It is always true that $U > U_{new}$.

Proof. Beginning with

$$L_{\text{CUR}} < R_{\text{CUR}}$$

and using the fact that $R_{\text{CUR}} - \Delta L_{\text{CUR}} > 0$, we have

$$\begin{align} -L_{\text{CUR}} &> -R_{\text{CUR}} \tag{33}\\ -L_{\text{CUR}}\Delta L_{\text{CUR}} &> -R_{\text{CUR}}\Delta L_{\text{CUR}} \tag{34}\\ L_{\text{CUR}}R_{\text{CUR}} - L_{\text{CUR}}\Delta L_{\text{CUR}} &> L_{\text{CUR}}R_{\text{CUR}} - R_{\text{CUR}}\Delta L_{\text{CUR}} \tag{35}\\ L_{\text{CUR}}(R_{\text{CUR}} - \Delta L_{\text{CUR}}) &> R_{\text{CUR}}(L_{\text{CUR}} - \Delta L_{\text{CUR}}) \tag{36}\\ & \tag{37}\end{align}$$

Dividing both sides by $R_{\text{CUR}}\,(R_{\text{CUR}} - \Delta L_{\text{CUR}})$ gives

$$\frac{L_{\text{CUR}}}{R_{\text{CUR}}} > \frac{L_{\text{CUR}} - \Delta L_{\text{CUR}}}{R_{\text{CUR}} - \Delta L_{\text{CUR}}}, \tag{38}$$

which means

$$U > U_{\text{new}}. \tag{39}$$

□

To recap, we have shown that when a loan is liquidated due to LTV (but not Loan Limit) breach, the utilization of the loan-limit improves (decreases). Therefore, if the global loan limit was respected before the loan liquidation, it will also be respected after the loan liquidation. **A loan liquidation due to LTV breach cannot lead to another loan liquidation due to global loan limit breach.**

Loan limit $\rightarrow$ LTV

This case reduces to a cascade of $(b_1 + 1)$ loan liquidation checks, since there is an initial Loan Limit breach in addition to the possible b_1 checks from subsequent LTV breach cascade, established in the LTV $\rightarrow$ LTV case.

Loan Limit $\rightarrow$ Loan Limit

The situation where one loan limit breach leads subsequent loan limit breaches is somewhat reminiscent of geometric progression. This is because each loan liquidation improves (decreases) the global loan limit utilization as we have already seen in Section 6. Therefore, such a cascade rapidly converges to a point where it ends.

The length of this cascade is strongly dependent on the size of the smallest possible loan relative to the CUR reserves. On the other hand, the lower the global loan limit λ, the shorter the cascade. The precise loan values needed for such cascades can be found by solving sets of inequalities.

Example. To how the systems of inequalities emerge in this scenario, consider the 3-step cascade which liquidates 3 loans one-by-one:

$$\frac{L^1_{\text{CUR}}}{R_{\text{CUR}}} + \frac{L^2_{\text{CUR}}}{R_{\text{CUR}}} + \frac{L^3_{\text{CUR}}}{R_{\text{CUR}}} > \lambda, \tag{40}$$

$$\frac{L^2_{\text{CUR}}}{R_{\text{CUR}}} + \frac{L^3_{\text{CUR}}}{R_{\text{CUR}}} \leq \lambda, \tag{41}$$

$$\frac{L^2_{\text{CUR}}}{R_{\text{CUR}}} + \frac{L^3_{\text{CUR}}}{R_{\text{CUR}}} > \lambda\left(1 - \frac{L^1_{\text{CUR}}}{R_{\text{CUR}}}\right), \tag{42}$$

$$\frac{L^3_{\text{CUR}}}{R_{\text{CUR}}} \leq \lambda\left(1 - \frac{L^1_{\text{CUR}}}{R_{\text{CUR}}}\right), \tag{43}$$

$$\frac{L^3_{\text{CUR}}}{R_{\text{CUR}}} > \lambda\left(1 - \frac{L^1_{\text{CUR}}}{R_{\text{CUR}}} - \frac{L^2_{\text{CUR}}}{R_{\text{CUR}}}\right), \tag{44}$$

$$L_{\text{min}} < \frac{L^1_{\text{CUR}}}{R_{\text{CUR}}} \leq \lambda, \tag{45}$$

$$L_{\text{min}} < \frac{L^2_{\text{CUR}}}{R_{\text{CUR}}} \leq \lambda, \tag{46}$$

$$L_{\text{min}} < \frac{L^3_{\text{CUR}}}{R_{\text{CUR}}} \leq \lambda. \tag{47}$$

Eq. 40 says that the loan limit is breached initially; Eq. 41 ensures a cascade processes loans one-by-one, as otherwise more than one loan would be liquidated to begin with. Eq. 42 makes sure that the cascade continues due to liquidity used up in the liquidation of the first loan; Eq. 43 again makes sure that only one loan (and not two) is liquidated at a time. Eq. 44 finishes the cascade with the liquidation of the last loan. The final three equations make sure that each of the individual loans are not too small (due to the protocol fee mechanism) and not too large (such that they do not exhaust the loan limit individually). Given a minimum loan amount $L_{\text{min}} = 10^{-2} R_{\text{CUR}}$, this system is solved by

$$\frac{L^1_{\text{CUR}}}{R_{\text{CUR}}} = 0.0976206, \tag{48}$$

$$\frac{L^2_{\text{CUR}}}{R_{\text{CUR}}} = 0.0104974, \tag{49}$$

$$\frac{L^3_{\text{CUR}}}{R_{\text{CUR}}} = 0.0915026. \tag{50}$$

In general, such cascades start from a large loan to be liquidated first, and then rapidly decrease in size.

The cascade length can also be increased by decreasing reserves due to LP burns. However, all the realistic settings include cascades that are much shorter than the worst LTV→LTV case of b_1 liquidation checks.

Practical Considerations of Liquidation Cascades

While it is possible to provide an exact bound on the number of ticks that need to be checked for each type of cascade, it is not in general guaranteed that only one such cascade will occur. It is theoretically possible for the loan tick structure to reach such a state that multiple consecutive liquidations occur, potentially even causing enough liquidations that the gas necessary for liquidation checks would cause a trade to revert due to insufficient gas.

While acknowledging that such a situation is theoretically possible, let us briefly address why it is unlikely in practice.

The Tick Structure Necessary for Long Cascades

As exemplified by the system of inequalities shown in the Loan Limit ⇒ Loan Limit case, a series of multiple liquidation cascades will require loans to meet many conditions both in terms of loan amount and tick placement. Such a precisely-placed series of loans is unlikely to occur naturally. However, it is theoretically possible that such a sequence could be constructed by an Adversary, motivated either by malice or by a desire to protect their own loans from liquidation.

An adversary hoping to create such a placement would need at least the following attributes:

1. sufficient sophistication to determine the precise structure needed to ensure the cascade,
2. sufficient capital resources to create the loans themselves, as well as to not be bothered by the loan origination fees.

Were an adversary to meet these conditions, the precise loan structure they create would be easily disrupted by the routine actions of other users. In particular, upward price moves from trades, loans created by other users on the Lending Market, and LPs adding liquidity to the DEX could all change the structure of liquidation cascades in ways beyond an Adversary's control.

Completing Trades

Even if an Adversary were successful in creating an obtrusive series of loans to cause a very long series of liquidation cascades, the trader still has the option of breaking the trade into smaller pieces that have less impact.

In the end, both theoretical and practical considerations ensure the system is unlikely to stay in a state with a large number of liquidation cascades, and can continue to function even if it does.