A Theory of Automated Market Makers in DeFi

Massimo Bartoletti, James Hsin-Yu Chiang, Alberto Lluch-Lafuente

A theory of Automated Market Makers in DeFi

Massimo Bartoletti[1][0000−0003−3796−9774], James Hsin-yu Chiang[2][0000−0002−5126−9494], and Alberto Lluch-Lafuente[2][0000−0001−7405−0818]

[1] Università degli Studi di Cagliari, Cagliari, Italy bart@unica.it
[2] Technical University of Denmark, DTU Compute, Copenhagen, Denmark
{jchi,albl}@dtu.dk

Abstract. Automated market makers (AMMs) are one of the most prominent decentralized finance (DeFi) applications. They allow users to exchange units of different types of crypto-assets, without the need to find a counter-party. There are several implementations and models for AMMs, featuring a variety of sophisticated economic mechanisms. We present a theory of AMMs. The core of our theory is an abstract operational model of the interactions between users and AMMs, which can be concretised by instantiating the economic mechanisms. We exploit our theory to formally prove a set of fundamental properties of AMMs, characterizing both structural and economic aspects. We do this by abstracting from the actual economic mechanisms used in implementations and identifying sufficient conditions which ensure the relevant properties. Notably, we devise a general solution to the *arbitrage problem*, the main game-theoretic foundation behind the economic mechanisms of AMMs.

1 Introduction

Decentralized finance (DeFi) is emerging as an alternative to the traditional finance, boosted by blockchain-based crypto-tokens and smart contracts. One of the main DeFi applications are *Automated Market Makers (AMMs)*, which allow users to exchange crypto-tokens of different types without the intermediation of third parties. As of April 2021, the two AMM platforms leading by user activity, Uniswap [14] and Curve Finance [5], alone hold $8.1B and $6.2B worth of tokens, and process $1.5B and $210M worth of transactions daily [4,12].

AMMs are inherently hard to design, implement and understand, since they involve sophisticated economic incentive mechanisms. Although they generally only expose a handful of callable functions, interactions with AMMs are sensitive to transaction ordering [21,23,28,31]: thus, actors with the power to influence the order of transactions in the blockchain may be incentivized to do so for profit or to harm specific users. Thus, there exists a need for foundational work to devise formal models of AMMs which allow the study of their fundamental properties, transaction concurrency and the effect of economic incentives.

Current descriptions of AMMs are either economic models [17–19,24], which focus on the efficacy of incentive design, or the actual implementations. While economic models are useful to understand the macroscopic financial aspects of

AMMs, they do not precisely describe the interactions between AMMs and their users. Still, understanding these interactions is crucial to determine possible deviations from the expected behaviour. Implementations, instead, reflect the exact behaviour of AMMs, but at a level of detail that hampers high-level understanding and reasoning. Moreover, the rich variety of implementations, proposals and models for AMMs, each featuring different sophisticated economic mechanisms, makes it difficult to establish comparisons between AMM designs or to provide a clear contour for the space of possible "well behaving" designs.

Contributions In this paper we address these challenges by developing a theory of AMMs. The core of our theory is a formal model of AMMs (§2), based on a thorough inspection of leading AMM implementations like Uniswap [13], Curve [16], and Balancer [3], as well as existing models from the literature [1,31]. Our model precisely describes the interactions between users and AMMs, and their main economic features. An original aspect of our model is that it is parametric with respect to the key economic mechanism — the *swap invariant* — that algorithmically determines exchange rates between tokens. This makes our model general enough to encompass the mainstream implementations and models of AMMs. With respect to economic models, our theory considers implementation details that are crucial to guarantee (efficient) computability in practice. Our model features an *executable semantics*, which can support implementations and analysis tools. As a matter of fact, an open-source Ocaml implementation of our executable semantics is provided as a companion of this paper.[3]

Building upon our model, we prove a set of properties characterizing both structural (§3) and economic (§4) aspects of AMMs. With respect to previous works, which focus on specific economic mechanisms, all our results are parametric with respect to swap invariants. We identify indeed, for each property, a set of conditions on swap invariants that are sufficient for the property to hold. Our results include fundamental structural properties such as *net worth preservation* ("value cannot be created/destroyed"), *liquidity* ("assets cannot be frozen within an AMM"), and *transaction concurrency* ("two transactions can be executed in any order"), as well as fundamental economic properties such as *incentive-consistency*, which ensures an incentive feedback loop between deposits and swaps of tokens. Most notably, we generalize the formulation and the solution to the so-called *arbitrage problem*, the main game-theoretic foundation behind the economic aspects of AMMs. We show that users are incentivized to perform actions that keep the swap rates aligned with the exchange rates given by price oracles. Namely, if an AMM offers a better swap rate than the oracles' exchange rate, rational users will perform swaps to narrow the gap. Further, we show that, under certain conditions, deposits and swaps incentivize each other.

Overall, our theory encompasses and generalizes the main functional and economic aspects of the mainstream AMM implementations, providing solid grounds for the design of future AMMs. Due to space constraints, we provide the proofs of our statements in a separate technical report [26].

[3] https://github.com/blockchain-unica/defi-workbench

2 A formal model of Automated Market Makers

We introduce a formal, operational model of AMMs, focussing on the common features implemented by the main AMM platforms. We discuss in §6 the differences between these platforms and our model.

2.1 AMM states

Basics We assume a set of ***users*** $\mathbb{A}$, ranged over by $\mathsf{A}, \mathsf{A}', \ldots$, and a set of ***token types*** $\mathbb{T}$, ranged over by $\tau, \tau', \ldots$. We denote with $\mathbb{T}_0 \subseteq \mathbb{T}$ a specific subset of token types that we call *initial* (they include, e.g., native blockchain tokens). The rest of the token types in $\mathbb{T}$ represent *minted* tokens, denoted as pairs (τ, τ') of distinct token types, and which represent shares in an AMM. We use v, v', r, r' to range over nonnegative real numbers ($\mathbb{R}_0^+$), and we write $r : \tau$ to denote r units of token type τ. We denote with $\mathrm{dom}\, f$ the domain of a partial map f. We model the ***wallet*** of a user A as a term $\mathsf{A}[\sigma]$, where the partial map $\sigma \in \mathbb{T} \rightharpoonup \mathbb{R}_0^+$ represents A's token holdings. We model an ***AMM*** as a pair of the form $(r_0 : \tau_0, r_1 : \tau_1)$, representing the fact that the AMM is holding, respectively, r_0 and r_1 units of token types τ_0 and τ_1.

States We formalise the interaction between users and AMMs as a labelled transition system (LTS). Its labels $\mathsf{T}, \mathsf{T}', \ldots$ represent blockchain ***transactions***, while the ***states*** $\Gamma, \Gamma', \ldots$ are compositions of wallets and AMMs:

$$\mathsf{A}_1[\sigma_1] \mid \cdots \mid \mathsf{A}_n[\sigma_n] \mid (r_1 : \tau_1, r_1' : \tau_1') \mid \cdots \mid (r_k : \tau_k, r_k' : \tau_k')$$

where all A_i are distinct, and for all $i \neq j$: $\tau_i \neq \tau_i'$ (i.e., the token types in an AMM are *distinct*), and $(\tau_i = \tau_j \Rightarrow \tau_i' \neq \tau_j') \wedge (\tau_i = \tau_j' \Rightarrow \tau_i' \neq \tau_j)$ (i.e., distinct AMMs cannot hold exactly the same token types). Two AMMs can indeed have a common token type τ, as in $(r_1 : \tau_1, r : \tau)$, $(r' : \tau, r_2' : \tau_2')$, thus enabling indirect trades between token pairs not directly provided by any AMM. A state Γ is *initial* when it only contains wallets with initial tokens. We treat states as sets of terms (wallets/AMMs): hence, Γ and Γ' are equivalent when they contain the same terms; for a term Q, we write $Q \in \Gamma$ when $\Gamma = Q \mid \Gamma'$, for some Γ'.

Example 1. Figure 1 shows an execution trace in our model, that we will explain in detail later in Example 5. We write $\Gamma \xrightarrow{\mathsf{T}} \Gamma'$ for a state transition from Γ to Γ', triggered by a transaction T. The first two states are initial, while the others contain an AMM for a token pair (τ_0, τ_1). □

Token supply We define the ***supply*** of a token type τ in a state Γ as the sum of the balances of τ in all the wallets and the AMMs occurring in Γ. Formally:

$$\mathit{sply}_\tau(\mathsf{A}[\sigma]) = \begin{cases} \sigma(\tau) & \text{if } \tau \in \mathrm{dom}\, \sigma \\ 0 & \text{otherwise} \end{cases} \qquad \mathit{sply}_\tau(r_0 : \tau_0, r_1 : \tau_1) = \begin{cases} r_i & \text{if } \tau = \tau_i \\ 0 & \text{otherwise} \end{cases}$$

$$\mathit{sply}_\tau(\Gamma \mid \Gamma') = \mathit{sply}_\tau(\Gamma) + \mathit{sply}_\tau(\Gamma')$$

$$\mathsf{A}[70:\tau_0, 80:\tau_1] \mid \mathsf{B}[30:\tau_0]$$

$$\xrightarrow{\mathsf{A}:\mathsf{xfer}(\mathsf{B},10:\tau_1)} \mathsf{A}[70:\tau_0, 70:\tau_1] \mid \mathsf{B}[30:\tau_0, 10:\tau_1] \tag{1}$$

$$\xrightarrow{\mathsf{A}:\mathsf{dep}(70:\tau_0,70:\tau_1)} \mathsf{A}[70:(\tau_0,\tau_1)] \mid \mathsf{B}[\cdots] \mid (70:\tau_0, 70:\tau_1) \tag{2}$$

$$\xrightarrow{\mathsf{B}:\mathsf{swapL}(30:\tau_0,20:\tau_1)} \mathsf{A}[\cdots] \mid \mathsf{B}[0:\tau_0, 31:\tau_1] \mid (100:\tau_0, 49:\tau_1) \tag{3}$$

$$\xrightarrow{\mathsf{B}:\mathsf{swapR}(29:\tau_0,21:\tau_1)} \mathsf{A}[\cdots] \mid \mathsf{B}[30:\tau_0, 10:\tau_1] \mid (70:\tau_0, 70:\tau_1) \tag{4}$$

$$\xrightarrow{\mathsf{B}:\mathsf{rdm}(30:(\tau_0,\tau_1))} \mathsf{A}[30:\tau_0, 30:\tau_1, 40:(\tau_0,\tau_1)] \mid \mathsf{B}[\cdots] \mid (40:\tau_0, 40:\tau_1) \tag{5}$$

$$\xrightarrow{\mathsf{B}:\mathsf{swapL}(30:\tau_0,16:\tau_1)} \mathsf{A}[\cdots] \mid \mathsf{B}[0:\tau_0, 27:\tau_1] \mid (70:\tau_0, 23:\tau_1) \tag{6}$$

$$\xrightarrow{\mathsf{A}:\mathsf{rdm}(30:(\tau_0,\tau_1))} \mathsf{A}[82:\tau_0, 47:\tau_1, 10:(\tau_0,\tau_1)] \mid \mathsf{B}[\cdots] \mid (18:\tau_0, 6:\tau_1) \tag{7}$$

Fig. 1: Interactions between two users and an AMM.

Example 2. Consider the first state in Figure 1, $\Gamma_1 = \mathsf{A}[70:\tau_0, 80:\tau_1] \mid \mathsf{B}[30:\tau_0]$. We have that $sply_{\tau_0}(\Gamma_1) = 70 + 30 = 100$, while $sply_{\tau_1}(\Gamma_1) = 80$. Observe that the supply of both token types remains constant in Figure 1; we will show in Lemma 2 that the supply of initial token types is always preserved. □

Token prices and net worth Assume that initial tokens are priced by a global oracle $P^0 \in \mathbb{T}_0 \to \mathbb{R}_0^+$. We then define the ***price*** $P_\tau(\Gamma)$ of a token $\tau \in \mathbb{T}$ (either initial or minted) in a state Γ inductively as follows:

$$\begin{aligned} P_\tau(\Gamma) &= P^0(\tau) && \text{if } \tau \in \mathbb{T}_0 \\ P_{(\tau_0,\tau_1)}(\Gamma) &= \frac{r_0 \cdot P_{\tau_0}(\Gamma) + r_1 \cdot P_{\tau_1}(\Gamma)}{sply_{(\tau_0,\tau_1)}(\Gamma)} && \text{if } (r_0:\tau_0, r_1:\tau_1) \in \Gamma \end{aligned} \tag{8}$$

The main idea is that initial tokens are priced using directly the global oracle while minted tokens are priced under the assumption that they can be redeemed. Their price, hence, is obtained by (recursively) calculating the price of the tokens that can be obtained by redeeming them (i.e. the proportions of the reserves r_0 and r_1 given the current supply). This intuition will be further formalized later in Lemma 6.

Example 3. Let $\Gamma_7 = \mathsf{A}[82:\tau_0, 47:\tau_1, 10:(\tau_0,\tau_1)] \mid \cdots$ be the final state in Figure 1. We have that $sply_{(\tau_0,\tau_1)}(\Gamma_7) = 10$. Assume that the prices of initial tokens are $P^0(\tau_0) = 5$ and $P^0(\tau_1) = 9$. The price of the minted token (τ_0,τ_1) is hence:

$$P_{(\tau_0,\tau_1)}(\Gamma_7) = \frac{1}{10}\Big(18 \cdot P_{\tau_0}(\Gamma_7) + 6 \cdot P_{\tau_1}(\Gamma_7)\Big) = \frac{18}{10} \cdot 5 + \frac{6}{10} \cdot 9 = 14.4 \qquad \square$$

We now define a key concept to understand the incentives for users to participate in AMMs, namely the ***net worth*** of a user A in a state Γ:

$$W_{\mathsf{A}}(\Gamma) \;=\; \begin{cases} \sum_{\tau \in \mathrm{dom}\,\sigma} \sigma(\tau) \cdot P_\tau(\Gamma) & \text{if } \mathsf{A}[\sigma] \in \Gamma \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The ***global net worth*** $W(\Gamma)$ of a state Γ is the sum of the net worth in users' wallets. The token units held in AMMs are not accounted for by $W(\Gamma)$, because their value is already recorded by minted tokens held in users' wallets. Indeed, the equality $sply_{(\tau_0,\tau_1)}(\Gamma) \cdot P_{(\tau_0,\tau_1)}(\Gamma) = r_0 \cdot P_{\tau_0}(\Gamma) + r_1 \cdot P_{\tau_1}(\Gamma)$ between the net value of a minted token and the value of the AMM is a direct consequence of the definition of price in (8).

As we shall see later, one of the main goals of users is to maximize their net worth. This can be achieved through different interactions with the AMM (e.g., by investing tokens or trading units of differently priced token types).

Example 4. Recall from Figure 1 the state $\Gamma_1 = \mathsf{A}[70 : \tau_0, 80 : \tau_1] \mid \mathsf{B}[30 : \tau_0]$, where τ_0 and τ_1 are initial tokens. Assume again that the prices are $P^0(\tau_0) = 5$ and $P^0(\tau_1) = 9$. The users' net worth in Γ_1 are then:

$$W_{\mathsf{A}}(\Gamma_1) = 70 \cdot P^0(\tau_0) + 80 \cdot P^0(\tau_1) = 1070 \qquad W_{\mathsf{B}}(\Gamma_1) = 30 \cdot P^0(\tau_0) = 150$$

In $\Gamma_7 = \mathsf{A}[82 : \tau_0, 47 : \tau_1, 10 : (\tau_0, \tau_1)] \mid \mathsf{B}[0 : \tau_0, 27 : \tau_1] \mid (18 : \tau_0, 6 : \tau_1)$ we have:

$$W_{\mathsf{A}}(\Gamma_7) = 82 \cdot P_{\tau_0}(\Gamma_7) + 47 \cdot P_{\tau_1}(\Gamma_7) + 10 \cdot P_{(\tau_0,\tau_1)}(\Gamma_7) = 977$$
$$W_{\mathsf{B}}(\Gamma_7) = 27 \cdot P_{\tau_1}(\Gamma_7) = 243$$

Note that the net worth of A has decreased w.r.t. the initial state, while the net worth of B has increased. One may think that B has been more successful than A, but this depends on the users' goals. Note, e.g., that A holds 10 units of the minted token (τ_0, τ_1), whose price may increase in the future. □

2.2 AMM semantics

We now formally describe the interactions of the AMM that give rise to state transitions. State transitions are triggered by the ***transactions*** in Table 1. We formalise below their behaviour, but we give before an overview of our running example from Figure 1.

Example 5. Figure 1 actually displays a sequence of transitions in the LTS of our model. To keep the example simple, we have used there the *constant product* swap invariant, which requires swap transactions to preserve the product between the amounts of the two tokens in the AMM; further, we have assumed no fees. In step (1), A transfers $10 : \tau_1$ from her wallet to B's. In step (2), A creates a new AMM, depositing $70 : \tau_0$ and $70 : \tau_1$; in return, she receives 70 units of the minted token (τ_0, τ_1). In step (3), B swaps 30 of his units of τ_0 for *at least* 20 units of τ_1. The actual amount of units of τ_1 received by B is 21: indeed, $(70 + 30) \cdot (70 - 21) = 70 \cdot 70$, hence 21 satisfies the constant product swap invariant. In step (4), B reverses his prior action by swapping 21 of his units of τ_1 for *at least* 29 units of τ_0. Here, the actual amount of units of τ_1 received by B is 30, which also satisfies the constant product swap invariant. In step (5), B redeems 30 units of the minted token (τ_0, τ_1), accordingly reducing the funds in the AMM. Note that the received tokens exhibit the same 1-to-1 ratio as in the

$\mathsf{A} : \mathsf{xfer}(\mathsf{B}, v : \tau)$	A transfers $v : \tau$ to B
$\mathsf{A} : \mathsf{dep}(v_0 : \tau_0, v_1 : \tau_1)$	A deposits $v_0 : \tau_0$ and $v_1 : \tau_1$ to an AMM $(r_0 : \tau_0, r_1 : \tau_1)$, receiving in return some units of the minted token (τ_0, τ_1)
$\mathsf{A} : \mathsf{swapL}(v_0 : \tau_0, v_1 : \tau_1)$	A tranfers $v_0 : \tau_0$ to an AMM $(r_0 : \tau_0, r_1 : \tau_1)$, receiving in return *at least* v_1 units of τ_1
$\mathsf{A} : \mathsf{swapR}(v_0 : \tau_0, v_1 : \tau_1)$	A tranfers $v_1 : \tau_1$ to an AMM $(r_0 : \tau_0, r_1 : \tau_1)$, receiving in return *at least* v_0 units of τ_0
$\mathsf{A} : \mathsf{rdm}(v : \tau)$	A redeems v units of minted token $\tau = (\tau_0, \tau_1)$ from an AMM $(r_0 : \tau_0, r_1 : \tau_1)$, receiving in return some units of τ_0 and τ_1

Table 1: AMM transactions.

initial deposit at step (2). In step (6), B swaps 30 of his units of τ_0 for *at least* 16 units of τ_1. Unlike in the previous swap at step (3), now the actual amount of τ_1 received by B is 17. Note that the implied *swap rate* between received τ_1 units and sent τ_0 units has deteriorated w.r.t. step (3), even if the pair (τ_0, τ_1) had the same 1-to-1 ratio of funds. This is caused by the reduction in funds resulting from A's redeem action: thus, the swap rate is sensitive to both the ratio of funds in the pair as well as their absolute balances, a key property of the incentive mechanisms, as we shall see later in §4. Finally, in step (7) A performs another redeem of 30 units of the minted token (τ_0, τ_1), thereby extracting 52 units of τ_0 and 17 units of τ_1 from the AMM. Note that the ratio of redeemed tokens is no longer 1-to-1 as in the previous redeem action (5), as the prior left swap has changed the ratio between the funds of τ_0 and τ_1 in the AMM. □

We now formalise the transition rules. We use the standard notation $\sigma\{^v/_x\}$ to update a partial map σ at point x: namely, $\sigma\{^v/_x\}(x) = v$, while $\sigma\{^v/_x\}(y) = \sigma(y)$ for $y \neq x$. Given a partial map $\sigma \in \mathbb{T} \rightharpoonup \mathbb{R}_0^+$, a token type $\tau \in \mathbb{T}$ and a partial operation $\circ \in \mathbb{R}_0^+ \times \mathbb{R}_0^+ \rightharpoonup \mathbb{R}_0^+$, we define the partial map $\sigma \circ v : \tau$ as follows:

$$\sigma \circ v : \tau = \begin{cases} \sigma\{^{\sigma(\tau) \circ v}/_\tau\} & \text{if } \tau \in \operatorname{dom}\sigma \text{ and } \sigma(\tau) \circ v \in \mathbb{R}_0^+ \\ \sigma\{^v/_\tau\} & \text{if } \tau \notin \operatorname{dom}\sigma \end{cases}$$

Token transfer A user A can transfer some of her tokens to another user B, provided that there are enough units of the token in A's wallet. Formally:

$$\frac{\sigma_\mathsf{A}(\tau) \geq v}{\mathsf{A}[\sigma_\mathsf{A}] \mid \mathsf{B}[\sigma_\mathsf{B}] \mid \Gamma \xrightarrow{\mathsf{A}:\mathsf{xfer}(\mathsf{B},v:\tau)} \mathsf{A}[\sigma_\mathsf{A} - v : \tau] \mid \mathsf{B}[\sigma_\mathsf{B} + v : \tau] \mid \Gamma} \text{ [XFER]}$$

A consequence of this rule is that tokens (both initial and minted) are *fungible*, i.e. individual units of the same token type are interchangeable. In particular, amounts of tokens of the same type can be split into smaller parts, and two amounts of tokens of the same type can be joined.

Deposit Any user can create an AMM for a token pair (τ_0, τ_1) provided that such an AMM is not already present in the state. This is achieved by the transaction $\mathsf{A} : \mathsf{dep}(v_0 : \tau_0, v_1 : \tau_1)$, through which A transfers $v_0 : \tau_0$ and $v_1 : \tau_1$ to

the new AMM. In return for the deposit, A receives a certain positive amount of units of a new token type (τ_0, τ_1), which is minted by the AMM. The exact amount of units received is irrelevant. In our model we choose v_0 but any other choice would be valid. We formalise this behaviour by the rule:

$$\frac{\sigma(\tau_i) \geq v_i > 0 \;\; (i \in \{0,1\}) \qquad \tau_0 \neq \tau_1 \qquad (\_ : \tau_0, \_ : \tau_1), (\_ : \tau_1, \_ : \tau_0) \notin \Gamma}{\begin{array}{l} \mathsf{A}[\sigma] \mid \Gamma \xrightarrow{\mathsf{A}:\mathsf{dep}(v_0:\tau_0, v_1:\tau_1)} \\ \mathsf{A}[\sigma - v_0 : \tau_0 - v_1 : \tau_1 + v_0 : (\tau_0, \tau_1)] \mid (v_0 : \tau_0, v_1 : \tau_1) \mid \Gamma \end{array}} \;\; [\textsc{Dep0}]$$

Once an AMM is created, any user can deposit tokens into it, as long as doing so preserves the ratio of the token holdings in the AMM. When a user deposits $v_0 : \tau_0$ and $v_1 : \tau_1$ to an existing AMM, it receives in return an amount of minted tokens of type (τ_0, τ_1). This amount is the ratio between the deposited amount v_0 and the ***redeem rate*** of (τ_0, τ_1) in the current state Γ, i.e. the ratio between the amount r_0 of τ_0 stored in the AMM, and the total supply $sply_{(\tau_0,\tau_1)}(\Gamma)$ of the minted token in the state.

$$\frac{\sigma(\tau_i) \geq v_i > 0 \;\; (i \in \{0,1\}) \qquad r_1 v_0 = r_0 v_1 \qquad v = \frac{v_0}{r_0} \cdot sply_{(\tau_0,\tau_1)}(\Gamma)}{\begin{array}{l} \Gamma \;=\; \mathsf{A}[\sigma] \;\mid\; (r_0 : \tau_0, r_1 : \tau_1) \;\mid\; \Gamma' \xrightarrow{\mathsf{A}:\mathsf{dep}(v_0:\tau_0, v_1:\tau_1)} \\ \quad \mathsf{A}[\sigma - v_0 : \tau_0 - v_1 : \tau_1 + v : (\tau_0, \tau_1)] \;\mid\; (r_0 + v_0 : \tau_0, r_1 + v_1 : \tau_1) \;\mid\; \Gamma' \end{array}} \;\; [\textsc{Dep}]$$

Note that the premise $r_1 v_0 = r_0 v_1$ ensures that the ratio between the holdings of τ_0 and τ_1 in the AMM is preserved by the dep transaction, i.e.:

$$\frac{r_1 + v_1}{r_0 + v_0} \;=\; \frac{r_1}{r_0}$$

As we shall see in §4, users are incentivized to invest tokens into AMMs by the fact that trading operations (i.e., swaps) are subject to a fee mechanism that makes the redeem rate increase over time.

Swap As shown in step (3) and on of Example 5, users can increase their net worth by swapping tokens. Any user A can swap units of τ_0 in her wallet for units of τ_1 in an AMM $(r_0 : \tau_0, r_1 : \tau_1)$ by firing a transaction $\mathsf{A} : \mathsf{swapL}(v_0 : \tau_0, v_1 : \tau_1)$. Here, v_0 is the amount of τ_0 transferred from A's wallet to the AMM, while v_1 is a *lower bound* on the amount of τ_1 that A will receive in return. The actual amount v is determined by a ***swap invariant*** $I \in \mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}_0^+$, that must hold between the amounts of τ_0 and τ_1 held in the AMM before and after the swap. To determine v, the AMM requires a fraction $0 < \phi \leq 1$ of v_0; the rest is considered as a fee (the parameter ϕ is the ***fee rate***). Formally:

$$\frac{\sigma(\tau_0) \geq v_0 > 0 \qquad I(r_0 + \phi\, v_0, r_1 - v) = I(r_0, r_1) \qquad 0 < v_1 \leq v \leq r_1}{\begin{array}{l} \mathsf{A}[\sigma] \;\mid\; (r_0 : \tau_0, r_1 : \tau_1) \;\mid\; \Gamma \xrightarrow{\mathsf{A}:\mathsf{swapL}(v_0:\tau_0, v_1:\tau_1)} \\ \mathsf{A}[\sigma - v_0 : \tau_0 + v : \tau_1] \;\mid\; (r_0 + v_0 : \tau_0, r_1 - v : \tau_1) \;\mid\; \Gamma \end{array}} \;\; [\textsc{SwapL}]$$

The effect of the fee is that the redeem rate of minted tokens increases; intuitively, the AMM retains a portion of the swapped amounts, but the overall

reserve is still distributed among all minted tokens, thereby ensuring liquidity (as we shall formally establish liquidity later on in Lemma 4).

Although actual AMM implementations use a variety of different swap invariants, with the common aim to incentivize users to perform swaps, all these invariants share a few common design choices. A crucial one is that there exists *exactly* one v which satisfies the equation in the premise of [SwapL]; further, swapping 0 units of τ_0 results in 0 units of τ_1. Formally, for all $r_0, r_1 > 0$:

$$\forall v \in \mathbb{R}_0^+ : \exists! v' \in \mathbb{R}_0^+ : I(r_0 + v, r_1 - v') = I(r_0, r_1) \tag{10}$$

Hereafter, we assume that I always respects this condition. A common swap invariant, implemented e.g. by Uniswap [13] and Mooniswap [7] (and also used in Example 5), is the *constant product invariant*, which requires that the product of the amounts of τ_0 and τ_1 in the AMM remains constant, i.e. $I(r_0, r_1) = r_0 \cdot r_1$.

The rule [SwapR] allows for swaps in the other direction:

$$\frac{\sigma(\tau_1) \geq v_1 > 0 \qquad I(r_0 - v, r_1 + \phi\, v_1) = I(r_0, r_1) \qquad 0 < v_0 \leq v \leq r_0}{\begin{array}{l}\mathsf{A}[\sigma] \mid (r_0 : \tau_0, r_1 : \tau_1) \mid \Gamma \xrightarrow{\mathsf{A}:\mathsf{swapR}(v_0:\tau_0, v_1:\tau_1)} \\ \mathsf{A}[\sigma + v : \tau_0 - v_1 : \tau_1] \mid (r_0 - v : \tau_0, r_1 + v_1 : \tau_1) \mid \Gamma\end{array}} \text{[SwapR]}$$

where we assume that I enjoys the "right" version of the condition (10).

It is worth explaining why the swap transactions specify lower bounds for the amount of return tokens, instead of an exact amount. In practice, when a user emits a transaction, she cannot predict the exact state in which the transaction will be actually committed. This makes it unfeasible to guess the exact amount that will preserve the swap invariant: hence, users can only specify a lower bound that they are willing to accept.

Redeem Any user can redeem units of a minted token (τ_0, τ_1), obtaining in return units of the underlying tokens τ_0 and τ_1. The redeemable amounts are determined by the redeem rate: each unit of (τ_0, τ_1) can be redeemed for equal fractions of τ_0 and τ_1 remaining in the AMM:

$$\frac{\sigma(\tau_0, \tau_1) \geq v > 0 \qquad v_0 = v \frac{r_0}{sply_{(\tau_0,\tau_1)}(\Gamma)} \qquad v_1 = v \frac{r_1}{sply_{(\tau_0,\tau_1)}(\Gamma)}}{\begin{array}{l}\Gamma = \mathsf{A}[\sigma] \mid (r_0 : \tau_0, r_1 : \tau_1) \mid \Gamma' \xrightarrow{\mathsf{A}:\mathsf{rdm}(v:(\tau_0,\tau_1))} \\ \mathsf{A}[\sigma + v_0 : \tau_0 + v_1 : \tau_1 - v : (\tau_0, \tau_1)] \mid (r_0 - v_0 : \tau_0, r_1 - v_1 : \tau_1) \mid \Gamma'\end{array}} \text{[Rdm]}$$

Example 6. Figure 2 shows the evolution of the AMM token holdings resulting from the trace in Figure 1, presented with Example 5. Recall that we have assumed a constant product swap invariant $x \cdot y = k$, and no swap fees ($\phi = 1$). We refer to a state in Figure 1 by the action number preceding it: the AMM $(70 : \tau_0, 70 : \tau_1)$ in state (2) is shown in Figure 2. Subsequent left (3) and right (4) swaps result in a traversal along $k = 70 \cdot 70$ from $(70 : \tau_0, 70 : \tau_1)$ to $(100 : \tau_0, 49 : \tau_1)$, and back as the swap invariant must hold for swap actions. The redeem action (5) reduces the holdings of both tokens by the same factor to reach $(40 : \tau_0, 40 : \tau_1)$. A left swap (6) traverses $k' = 40 \cdot 40$ to reach $(70 : \tau_0, 23 : \tau_1)$ in state (6), which is then followed by another redeem (7) action, reducing both token holdings proportionally to $(18 : \tau_0, 6 : \tau_1)$.

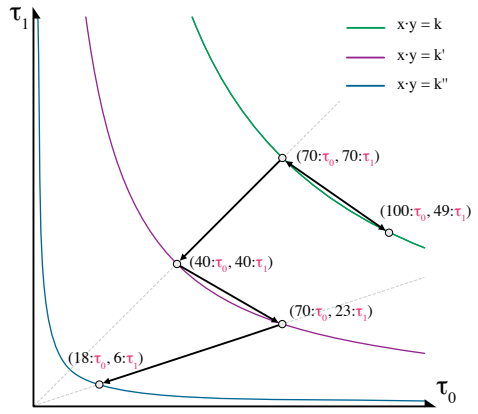

Fig. 2: Evolution of balances of AMM (τ_0, τ_1) along the trace in Figure 1.

3 Structural properties of AMMs

We now establish some structural properties of AMMs, which do not depend on the design of the economic mechanisms, i.e. on the choice of the swap invariant. We denote with $\to^*$ the reflexive and transitive closure of $\to$. Given a finite sequence of transactions $\lambda = \mathsf{T}_1 \cdots \mathsf{T}_k$, we write $\Gamma \xrightarrow{\lambda} \Gamma'$ when $\Gamma \xrightarrow{\mathsf{T}_1} \cdots \xrightarrow{\mathsf{T}_k} \Gamma'$. We say that a state Γ is *reachable* if $\Gamma_0 \to^* \Gamma$ for some initial Γ_0. We denote with $type(\mathsf{T})$ the type of T (i.e., $\mathsf{xfer}, \mathsf{dep}, \ldots$), with $wal(\mathsf{T})$ the set of wallets affected by T (e.g., $wal(\mathsf{A} : \mathsf{xfer}(\mathsf{B}, v : \tau)) = \{\mathsf{A}, \mathsf{B}\}$), and with $tok(\mathsf{T})$ the set of token types affected by T (e.g., $tok(\mathsf{A} : \mathsf{swapL}(v_0 : \tau_0, v_1 : \tau_1)) = \{\tau_0, \tau_1\}$).

First, we establish that the AMMs' LTS is deterministic. Note that, in swap rules, an unconstrained swap invariant I could admit different solutions to the equation in the premise: determinism is ensured by condition (10), which we assume to be true for all swap invariants.

Lemma 1 (Determinism). *If $\Gamma \xrightarrow{\mathsf{T}} \Gamma'$ and $\Gamma \xrightarrow{\mathsf{T}} \Gamma''$, then $\Gamma' = \Gamma''$.*

We can lift the statement to sequences of transactions by using a simple inductive argument. The same applies to other single-step results in this section.

Lemma 2 ensures that the supply of each *initial* token type τ is preserved by transitions (of any type). Note that preservation does not hold for *minted* tokens, as they can be created (by rule [Dep]) and destroyed (by rule [Rdm]).

Lemma 2. *For all $\tau \in \mathbb{T}_0$, if $\Gamma \to \Gamma'$ then $sply_\tau(\Gamma) = sply_\tau(\Gamma')$.*

Lemma 3 ensures that the *global* net worth is preserved by transactions, whereas the user's net worth is preserved only by redeems/deposits.

Lemma 3 (Preservation of net worth). *Let $\Gamma \xrightarrow{\mathsf{T}} \Gamma'$. Then, $W(\Gamma) = W(\Gamma')$. Further, if $type(\mathsf{T}) \in \{\mathsf{dep}, \mathsf{rdm}\}$ or $\mathsf{A} \notin wal(\mathsf{T})$, then $W_\mathsf{A}(\Gamma) = W_\mathsf{A}(\Gamma')$.*

Lemma 4 ensures that funds cannot be *frozen* in an AMM, i.e. that users can always redeem arbitrary amounts of the tokens deposited in an AMM.

Lemma 4 (Liquidity). *Let Γ be a reachable state such that $(r_0 : \tau_0, r_1 : \tau_1) \in \Gamma$ with $r_0 + r_1 > 0$. Then: (a) $sply_{(\tau_0,\tau_1)}(\Gamma) > 0$; (b) for all $r_0' \leq r_0$, there exists $r_1' \leq r_1$ such that $\Gamma \to^* (r_0' : \tau_0, r_1' : \tau_1) \mid \cdots$; (c) for all $r_1' \leq r_1$, there exists $r_0' \leq r_0$ such that $\Gamma \to^* (r_0' : \tau_0, r_1' : \tau_1) \mid \cdots$.*

We now study the concurrency of transactions. Two finite sequences of transactions λ_0 and λ_1 are *observationally equivalent*, in denoted $\lambda_0 \sim \lambda_1$, when, for all states Γ, if $\Gamma \xrightarrow{\lambda_0} \Gamma_0$ and $\Gamma \xrightarrow{\lambda_1} \Gamma_1$ then $\Gamma_0 = \Gamma_1$. We say that two distinct transactions T, T' are *concurrent* (denoted, $\mathsf{T} \# \mathsf{T}'$) if $\mathsf{T}\mathsf{T}' \sim \mathsf{T}'\mathsf{T}$. Note that this does not mean that T and T' cannot disable each other as demanded by stricter notions of concurrency. Lemma 5 provides sufficient conditions for two transactions to be concurrent: intuitively, two non-swap transactions are always concurrent, while swap transactions are concurrent with xfer transactions, and with any transactions which do not affect the same token types.

Lemma 5. *Two distinct transactions T_0, T_1 are concurrent if, for $i \in \{0,1\}$, $type(\mathsf{T}_i) \in \{\mathsf{swapL}, \mathsf{swapR}\}$ implies $tok(\mathsf{T}_i) \cap tok(\mathsf{T}_{1-i}) = \emptyset$ or $type(\mathsf{T}_{1-i}) = \mathsf{xfer}$.*

As we shall see later in §4, it is actually desirable, and crucial for the economic mechanism of AMMs, that swap transactions interfere with other transactions that trade the same token type.

The theory of Mazurkiewicz's trace languages [27] allows us to lift Lemma 5 to sequences of transactions. Let R be a symmetric and irreflexive relation on the set $\mathbb{X}$ of all transactions. The *Mazurkiewicz equivalence* $\sim_R$ is the least congruence in the free monoid $\mathbb{X}^*$ such that: $\forall \mathsf{T}, \mathsf{T}' \in \mathbb{X}\colon\ \mathsf{T}\, R\, \mathsf{T}' \implies \mathsf{T}\mathsf{T}' \sim_R \mathsf{T}'\mathsf{T}$. Theorem 1 states that the Mazurkiewicz equivalence constructed on the concurrency relation $\#$ is an observational equivalence.

Theorem 1 (Concurrent transactions can be reordered). $\sim_\# \subseteq \sim$.

A direct consequence of Theorem 1 is that we can transform a finite sequence of transactions into an observationally equivalent one by repeatedly exchanging adjacent concurrent transactions — provided that both sequences are executable in the LTS. For example, sequences of $\mathsf{A} : \mathsf{rdm}(\_)$ transactions can be freely reordered, resulting in the same, unique state. This is exploited in the following lemma, which supports the inductive definition of the price of minted tokens in (8): indeed, computing the net worth of a user A under that price definition corresponds to making A first redeem all her minted tokens, and then summing the price of the resulting initial tokens.

Lemma 6. *For all states Γ and users A, let $rdm_{\mathsf{A}}(\Gamma)$ be the unique state reached from Γ by performing only $\mathsf{A} : \mathsf{rdm}(\_)$ actions, such that A's wallet in $rdm_{\mathsf{A}}(\Gamma)$, only contains initial tokens. Then:*

$$W_{\mathsf{A}}(\Gamma) \;=\; \textstyle\sum_{\tau \in \mathrm{dom}\,\sigma} \sigma(\tau) \cdot P^0(\tau) \qquad \text{if } \mathsf{A}[\sigma] \in rdm_{\mathsf{A}}(\Gamma)$$

Example 7. Recall from Example 4 that $W_{\mathsf{A}}(\Gamma_7) = 977$. Assume that A performs a further transaction to redeem all 10 units of (τ_0, τ_1) from her wallet. The resulting state is $\Gamma_8 = \mathsf{A}[100 : \tau_0, 53 : \tau_1] \mid \cdots$. We compute A's net worth in that state, using the oracle token prices: $W_{\mathsf{A}}(\Gamma_8) = 100 \cdot P_{(\tau_0)}(\Gamma_7) + 53 \cdot P_{(\tau_1)}(\Gamma_7) = 100 \cdot 5 + 53 \cdot 9 = 977$, as correctly predicted by Lemma 6. □

4 Properties of AMM incentives

We now study the incentive mechanisms of AMMs. We start in §4.1 by introducing a few notions of *exchange rate*, which are pivotal to understanding these mechanisms. In §4.2 we devise general conditions on swap invariants, overall named *incentive-consistency*, which guarantee that AMMs enjoy relevant economic properties. In §4.3 we study solutions to the *arbitrage problem*, which is the key to incentivize users to perform swap operations towards an ideal state where the AMM's exchange rates align with the exchange rates set by price oracles. Finally, in §4.4 we study the incentives to swap and deposit larger amounts.

4.1 Exchange rates

The ***exchange rate*** between two token types is the number of units of one token needed to buy one unit of the other token at the current price. We define *L*eft and *R*ight versions of this notion, that reflect the direction of the exchange:

$$XL_\Gamma(\tau_0, \tau_1) \;=\; P_{\tau_0}(\Gamma)/P_{\tau_1}(\Gamma) \qquad XR_\Gamma(\tau_0, \tau_1) \;=\; P_{\tau_1}(\Gamma)/P_{\tau_0}(\Gamma) \tag{11}$$

The ***swap rate*** between τ_0 and τ_1 upon a payment of $v_i : \tau_i$ (for $i \in \{0, 1\}$) is the ratio between v and v_i, where v is the received amount of τ_{1-i} resulting from a swap action on an AMM $(r_0 : \tau_0, r_1 : \tau_1)$. We first introduce an auxiliary notion, parameterized over the balances r_0 and r_1, instead of the token types:

$$\begin{aligned} XL^{\mathsf{swap}}_\phi(v_0, r_0, r_1) \;&=\; v/v_0 \quad \text{if } I(r_0, r_1) = I(r_0 + \phi v_0, r_1 - v) \\ XR^{\mathsf{swap}}_\phi(v_1, r_0, r_1) \;&=\; v/v_1 \quad \text{if } I(r_0, r_1) = I(r_0 - v, r_1 + \phi v_1) \end{aligned} \tag{12}$$

The swap rate is parameterized over the fee rate ϕ: the case where $\phi = 1$ represents an ideal scenario with no fees: in this case, we write just $XL^{\mathsf{swap}}(v_0, r_0, r_1)$. We define the swap rate in a state Γ such that $(r_0 : \tau_0, r_1 : \tau_1) \in \Gamma$ as follows:

$$XL^{\mathsf{swap}}_{\Gamma,\phi}(v_0, \tau_0, \tau_1) = XL^{\mathsf{swap}}_\phi(v_0, r_0, r_1) \qquad XR^{\mathsf{swap}}_{\Gamma,\phi}(v_1, \tau_0, \tau_1) = XR^{\mathsf{swap}}_\phi(v_1, r_0, r_1)$$

We also define the ***redeem rate***. The left version is:

$$XL^{\mathsf{rdm}}_\Gamma(\tau_0, \tau_1) = r_0/sply_{(\tau_0,\tau_1)}(\Gamma) \qquad \text{if } (r_0 : \tau_0, r_1 : \tau_1) \in \Gamma \tag{13}$$

4.2 General properties of swap invariants

We now introduce a set of properties of swap invariants, called cumulatively ***incentive-consistency***, which overall incentivize users to interact with AMMs by performing swap and deposit actions.

Swap-rate continuity This property requires that, for all $r_0, r_1 > 0$:

$$\lim_{\varepsilon \to 0} XL^{\text{swap}}(\varepsilon, r_0, r_1) \;=\; 1/\lim_{\varepsilon \to 0} XR^{\text{swap}}(\varepsilon, r_0, r_1) \in \mathbb{R}^+ \tag{14}$$

Figure 3 (left) illustrates this property, displaying the points (x, y) which satisfy the constant product invariant $x \cdot y = k$. The left swap rate limit for the constant product invariant and $\phi = 1$ is $\lim_{\varepsilon \to 0} XL^{\text{swap}}(\varepsilon, r_0, r_1) = r_1/r_0$, while for the right swap we have $\lim_{\varepsilon \to 0} XR^{\text{swap}}(\varepsilon, r_0, r_1) = r_0/r_1$. Coinciding left swap limit and right swap limit inverse are illustrated as the slope of the product constant curve at a selected point in Figure 3 (left). The constant product invariant satisfies (14), i.e. it is swap-rate continuous.

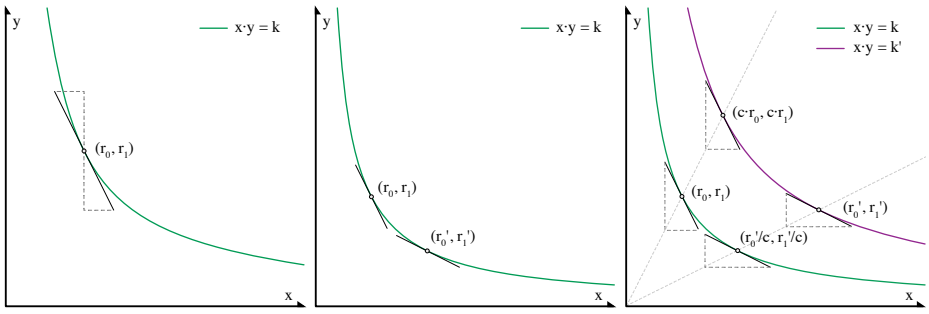

Fig. 3: The constant product invariant $I(x, y) = x \cdot y$ is swap-rate-consistent (left), demand-sensitive (center), non-depletable, funds-consistent (right) and swap-rate consistent (right).

Demand-sensitivity A swap invariant is demand-sensitive if the swap rate strictly decreases with demand. Formally, for all $r_0, r_1, r_0', r_1' > 0$:

$$I(r_0, r_1) = I(r_0', r_1') \wedge r_0' > r_0 \implies \lim_{\varepsilon \to 0} XL_\phi^{\text{swap}}(\varepsilon, r_0, r_1) \;>\; \lim_{\varepsilon \to 0} XL_\phi^{\text{swap}}(\varepsilon, r_0', r_1') \tag{15}$$

We implicitly require that (15) and the subsequent properties stated for the left version of an exchange rate also hold for the right version.

Figure 3 (center) depicts two points (r_0, r_1), (r_0', r_1') on the constant product curve, which satisfy $x \cdot y = k$ for identical k. For the constant product invariant, the left swap limit can be expressed as $\lim_{\varepsilon \to 0} XL^{\text{swap}}(\varepsilon, r_0, r_1) = \phi \cdot r_1/r_0$. For the given k and points in Figure 3 (center):

$$\lim_{\varepsilon \to 0} XL_\phi^{\text{swap}}(\varepsilon, r_0, r_1) = \phi \cdot k/r_0^2 \qquad \lim_{\varepsilon \to 0} XL_\phi^{\text{swap}}(\varepsilon, r_0', r_1') = \phi \cdot k/{r_0'}^2$$

Thus for $r_0' > r_0$, $\lim_{\varepsilon \to 0} XL_\phi^{\text{swap}}(\varepsilon, r_0, r_1) > \lim_{\varepsilon \to 0} XL_\phi^{\text{swap}}(\varepsilon, r_0', r_1')$: the constant product invariant is demand-sensitive.

Non-depletion This property ensures that the balance of tokens within an AMM cannot be zeroed via swaps. Formally, I is non-depletable when, for all $r_0, r_1 > 0$ and $r'_0, r'_1 \geq 0$:

$$I(r_0, r_1) = I(r'_0, r'_1) \implies r'_0, r'_1 \neq 0 \tag{16}$$

Note that the constant product invariant trivially satisfies this property.

Funds-consistency Deposits to an AMM ensure higher swap rates for a given input amount v, whereas redeems will reduce the swap rates for v. This behaviour is formalized later on in Theorem 4, but is a consequence of the funds-consistency property of the swap invariant. Formally, we require that for all $r_0, r_1, r'_0, r'_1 > 0$:

$$\begin{aligned} &I(r_0, r_1) \neq I(r'_0, r'_1) \iff \\ &\qquad \exists! c \in \mathbb{R}^+ \setminus \{1\} : I(c \cdot r'_0, c \cdot r'_1) = I(r_0, r_1) \ \wedge \ I(\tfrac{r_0}{c}, \tfrac{r_1}{c}) = I(r'_0, r'_1) \end{aligned} \tag{17}$$

Figure 3 (right) illustrates funds-consistency for the constant product invariant. Here, $r_0 \cdot r_1 = k \neq k' = r'_0 \cdot r'_1$. We observe that there exists a unique $c > 0$ in $(c \cdot r_0) \cdot (c \cdot r_1) = r'_0 \cdot r'_1 = k'$: namely, $c = \sqrt{(r'_0 \cdot r'_1)/(r_0 \cdot r_1)}$. Conversely, $(r'_0/c) \cdot (r'_1/c) = r_0 \cdot r_1$, which holds for the same value of c.

Swap-rate-consistency The design of AMMs aims to ensure that redeems and deposits do not interfere with the alignment of the swap rate towards the exchange rate. Since both deposits and redeems preserve the balance ratio of a token pair, we require swap rate limits for all balances of a given ratio to be constant. For all $r_0, r_1, c > 0$:

$$\lim_{\varepsilon \to 0} XL^{\mathsf{swap}}_{\phi}(\varepsilon, r_0, r_1) \ = \ \lim_{\varepsilon \to 0} XL^{\mathsf{swap}}_{\phi}(\varepsilon, c \cdot r_0, c \cdot r_1) \tag{18}$$

Figure 3 (right) illustrates equal swap rate limits for given r_0, r_1 and $c \cdot r_0, c \cdot r_1$. Here, $\lim_{\varepsilon \to 0} XL^{\mathsf{swap}}(\varepsilon, r_0, r_1) = \phi \cdot r_1 / r_0 = \phi \cdot (c \cdot r_1)/(c \cdot r_0)$ for $c > 0$.

Finally, the following lemma establishes that the constant product swap invariant (the one used e.g. by Uniswap and Mooniswap) is indeed incentive-consistent. We conjecture that the same is true for the swap invariants implemented by the other mainstream AMM platforms.

Lemma 7. *The constant product swap invariant is incentive-consistent.*

4.3 The arbitrage game

We now study the incentive mechanisms of AMMs from a game-theoretic perspective. Indeed, AMMs can be seen as multi-player games where users collaborate or compete to achieve possibly conflicting goals. In such games the allowed moves of users are the interactions with other users and with AMMs, while their goal is typically to increase their net worth.

The ***arbitrage problem*** is an interesting example of an AMM game since it is directly linked to the incentive of swaps in a way that makes AMMs track exchange rates. The arbitrage problem has been formalized for specific swap

invariants, namely the *weighted* and *constant product* swap invariant [17,19]. We generalize here the arbitrage problem to *arbitrary* swap invariants. We provide sufficient conditions for the existence of solutions, and we link the solutions to the expected relation between AMMs and exchange rates.

We model the arbitrage problem as a single-player, single-round game. The *initial game state* is $\Gamma_0 = \mathsf{A}[\sigma] \mid (r_1 : \tau_0, r_1 : \tau_1)$, where A is the only *player*. The *moves* of A are all the possible transactions fired by A; we also consider doing nothing as a possible move. The *goal* of A is to maximize her net worth, i.e. to maximize $W_{\mathsf{A}}(\Gamma) - W_{\mathsf{A}}(\Gamma_0)$, where Γ is the state resulting from executing the selected move. A *solution* to the game is a move that satisfies the goal, i.e. one of the optimal moves. We further assume that A holds no minted tokens containing (τ_0, τ_1) as a subterm (i.e., (τ_0, τ_1) itself, $((\tau_0, \tau_1), \tau_2)$, etc.). In this way, any change in A's net worth only depends on the exchange rate between τ_0 and τ_1, and on the transfer of value resulting from A's move.

Before presenting the solution to the game we examine the potential candidates for the solution. First, note that transfers are not valid solutions, as they can only decrease A's net worth. A second observation is that doing nothing, depositing or redeeming do not alter A's net worth (cf. Lemma 3). Hence, if one of such moves is a solution, so are the other two. The only moves that may affect A's net worth are swaps. For a swap to be a solution to the game, it must, first of all, result in a positive change of A's net worth. This happens when the swap rate is greater than the exchange rate. Theorem 2 presents the solution to the game. Note that if $\mathsf{swapL}(v_0 : \tau_0, v_1 : \tau_1)$ is a solution, then for all $v_1' \leq v_1$, also $\mathsf{swapL}(v_0 : \tau_0, v_1' : \tau_1)$ is a solution. Without loss of generality, our statement singles our the solution with the greatest v_1 (similarly for the right swap).

Theorem 2. *Let I be demand-sensitive and non-depletable, and let the initial state of the game be $\Gamma_0 = \mathsf{A}[\sigma] \mid (r_0 : \tau_0, r_1 : \tau_1)$, with $r_0, r_1 > 0$. Let $\sigma(\tau_0)$, $\sigma(\tau_1)$ be large enough to enable any needed swap. Then, the solution to the game is:*

- $\mathsf{A} : \mathsf{swapL}(v_0 : \tau_0, v_1 : \tau_1)$ *if* $\Gamma_0 \xrightarrow{\mathsf{A}:\mathsf{swapL}(v_0:\tau_0, v_1:\tau_1)} \Gamma$, *and:*

  (1) $\lim_{\varepsilon \to 0} XL^{\mathsf{swap}}_{\Gamma_0,\phi}(\varepsilon, \tau_0, \tau_1) > XL_{\Gamma_0}(\tau_0, \tau_1)$

  (2) $\lim_{\varepsilon \to 0} XL^{\mathsf{swap}}_{\phi}(\varepsilon, r_0 + \phi \cdot v_0, r_1 - v_1) = XL_{\Gamma}(\tau_0, \tau_1)$ *where* $\exists!\, \delta$:
  $I(r_0, r_1) = I(r_0 + \phi \cdot v_0, r_1 - v_1) = I(r_0 + \phi \cdot (v_0 + \varepsilon), r_1 - (v_1 + \delta))$

- $\mathsf{A} : \mathsf{swapR}(v_0 : \tau_0, v_1 : \tau_1)$ *if* $\Gamma_0 \xrightarrow{\mathsf{A}:\mathsf{swapR}(v_0:\tau_0, v_1:\tau_1)} \Gamma$, *and:*

  (1) $\lim_{\varepsilon \to 0} XR^{\mathsf{swap}}_{\Gamma_0,\phi}(\varepsilon, \tau_0, \tau_1) > XR_{\Gamma_0}(\tau_0, \tau_1)$

  (2) $\lim_{\varepsilon \to 0} XR^{\mathsf{swap}}_{\phi}(\varepsilon, r_0 - v_0, r_1 + \phi \cdot v_1) = XR_{\Gamma}(\tau_0, \tau_1)$ *where* $\exists!\, \delta$:
  $I(r_0, r_1) = I(r_0 - v_0, r_1 + \phi \cdot v_1) = I(r_0 - (v_0 + \delta), r_1 + \phi \cdot (v_1 + \varepsilon))$

- *do nothing (or do any deposit or redeem), otherwise.*

Intuitively, condition (1) requires that the swap rate for infinitesimal amounts is greater than the exchange rate in the initial state; (2) requires that in the state Γ reached by performing the move of the solution, the swap rate for infinitesimal amounts tends to the exchange rate — thus achieving one of the main desiderata on AMMs. Note that Γ is an *equilibrium*: no move from there can improve A's net worth, i.e. doing nothing is a solution for the arbitrage problem in Γ.

Note that for the swapL/swapR solutions, the swapped amounts are unique: this is a consequence on the assumption (10). An implicit desideratum on these solutions is that, given a specific instance of the swap invariant, they are efficiently computable: this is the case, e.g., for the constant product invariant [17].

For $\phi = 1$ we can observe by inspection of (14) that the do-nothing solution for Theorem 2 only holds for:

$$\lim_{\varepsilon \to 0} XL^{\mathsf{swap}}_{\Gamma_0}(\varepsilon, \tau_0, \tau_1) \;=\; 1/\lim_{\varepsilon \to 0} XR^{\mathsf{swap}}_{\Gamma_0}(\varepsilon, \tau_0, \tau_1) \;=\; XL_{\Gamma_0}(\tau_0, \tau_1) \tag{19}$$

Thus, the solution to the game results in the AMM tracking global exchange rates precisely: any infinitesimal deviation of the global exchange rate implies a swap action in the arbitrage game.

The assumption that the players' wallets are sufficiently large is common in formulations of the arbitrage problem. We note that any rational agent is incentivized to perform such a swap: the optimal solution to the arbitrage game can thus be approximated by multiple users exchanging smaller swap amounts. Furthermore, the availability of flash-loans [29, 30] can provide up-front funds, and thus significantly reduce the balance requirements for arbitrage swaps.

Finally, we prove that a AMM deposits and redeems do not affect the solution type of the arbitrage game. If the arbitrage solution prior to a deposit or redeem is swapL, swapR or nothing, the arbitrage solution in the subsequent state should remain of the same type.

Theorem 3. *Let I be incentive-consistent. Let $(r_0 : \tau_0, r_1 : \tau_1) \in \Gamma$ with $r_0, r_1 > 0$. If $\Gamma \xrightarrow{\mathsf{T}} \Gamma'$, type$(\mathsf{T}) \in \{\mathsf{dep}, \mathsf{rdm}\}$, then the arbitrage solutions in Γ and Γ' will have the same type or both be nothing.*

In other words, the design of AMMs aims to ensure that deposits and redeems do not interfere with the alignment of the swap rate towards the exchange rate.

Example 8. Consider the arbitrage game with player B and initial state $\Gamma_7 = \mathsf{B}[0 : \tau_0, 27 : \tau_1] \mid (18 : \tau_0, 6 : \tau_1) \mid \cdots$ resulting after the last step in Figure 1. Assuming the constant product invariant and no fees (i.e., $\phi = 1$), we have that:

$$\lim_{\varepsilon \to 0} XL^{\mathsf{swap}}_{\Gamma_7}(\varepsilon, \tau_0, \tau_1) \;=\; r_1/r_0 \;=\; 6/18 \;<\; 5/9 \;=\; XL_{\Gamma_7}(\tau_0, \tau_1)$$

$$\lim_{\varepsilon \to 0} XR^{\mathsf{swap}}_{\Gamma_7}(\varepsilon, \tau_0, \tau_1) \;=\; r_0/r_1 \;=\; 18/6 \;>\; 9/5 \;=\; XR_{\Gamma_7}(\tau_0, \tau_1)$$

Hence, by Theorem 2 it follows that the optimal move is $\mathsf{swapR}(v_0 : \tau_0, v_1 : \tau_1)$, for suitable v_0 and v_1. To find these values, we must solve for v_0 and v_1 the equations in item (2) of Theorem 2, i.e.:

$$\lim_{\varepsilon \to 0} XR^{\mathsf{swap}}_{\phi}(\varepsilon, r_0 - v, r_1 + v_1) = XR_{\Gamma}(\tau_0, \tau_1) \qquad I(r_0, r_1) = I(r_0 - v_0, r_1 + v_1)$$

Solving these equations gives:

$$v_1 = \sqrt{\frac{5}{9} \cdot r_0 r_1} - r_1 \approx 1.74 \qquad v_0 = \frac{r_0 v_1}{r_1 + v_1} \approx 4$$

By performing $\mathsf{swapR}(v_0 : \tau_0, v_1 : \tau_1)$ with these values from Γ_7, we obtain:

$$\Gamma \;=\; \mathsf{B}[4 : \tau_0, 25.26 : \tau_1] \mid (14 : \tau_0, 7.74 : \tau_1) \mid \cdots$$

This action maximizes B's net worth: indeed, we have $W_{\mathsf{B}}(\Gamma_7) = 243$ and $W_{\mathsf{B}}(\Gamma) = 247.6$; any other action will result in a lower net worth for B. □

4.4 Incentivizing deposits and swaps

Theorem 2 ensures that incentive-consistent AMMs incentivize swaps to align to exchange rates. We now show that, under certain conditions, deposits and swaps incentivize each other. The intuition is that larger amounts of tokens in an AMM provide better swap rates, therefore attracting users interested in swaps. These swaps, in turn, result in increased redeem rates, making the AMM attractive for further deposits. Note that this behaviour relies on an underlying assumption of our model, i.e. that exchange rates are stable: oracle prices are fixed. In the wild, exchange rates can vary over time, possibly making the net worth of users holding minted AMM tokens decrease: this phenomenon is commonly referred to as *impermanent loss* [9].

The following theorem shows that deposits increase swap rates, hence incentivizing swaps, whilst redeems have the opposite effect.

Theorem 4. *Let Γ be incentive-consistent. Let $\Gamma = (r_0 : \tau_0, r_1 : \tau_1) \mid \cdots$, with $r_0, r_1 > 0$, and let $\Gamma \xrightarrow{\mathsf{A}:\ell} \Gamma'$. Then, for all $v \in \mathbb{R}^+$:*

$$\begin{matrix} XL^{\mathsf{swap}}_{\Gamma,\phi}(v, \tau_0, \tau_1) \circ XL^{\mathsf{swap}}_{\Gamma',\phi}(v, \tau_0, \tau_1) \\ XR^{\mathsf{swap}}_{\Gamma,\phi}(v, \tau_0, \tau_1) \circ XR^{\mathsf{swap}}_{\Gamma',\phi}(v, \tau_0, \tau_1) \end{matrix} \quad \textit{where } \circ = \begin{cases} < & \textit{if } \ell = \mathsf{dep}(\_ : \tau_0, \_ : \tau_1) \\ > & \textit{if } \ell = \mathsf{rdm}(\_ : (\tau_0, \tau_1)) \end{cases}$$

We now show that, under certain conditions, swaps incentivize deposits. Intuitively, swaps contribute to higher redeem rates, which increase the net wealth of the holders of minted AMM tokens:

Theorem 5. *Let Γ be incentive-consistent. Let $\Gamma = (r_0 : \tau_0, r_1 : \tau_1) \mid \cdots$ and $\Gamma \to^* \Gamma'$, where $\Gamma' = (r'_0 : \tau_0, r'_1 : \tau_1) \mid \cdots$. If $r_1/r_0 = r'_1/r'_0$ then:*

$$XL^{\mathsf{rdm}}_{\Gamma}(\tau_0, \tau_1) \leq XL^{\mathsf{rdm}}_{\Gamma'}(\tau_0, \tau_1) \qquad XR^{\mathsf{rdm}}_{\Gamma}(\tau_0, \tau_1) \leq XR^{\mathsf{rdm}}_{\Gamma'}(\tau_0, \tau_1)$$

Recall that a user who deposits into an AMM $(r_0 : \tau_0, r_1 : \tau_1)$ in state Γ receives in return an amount of minted tokens. A consequence of Theorem 5 is that these minted tokens can be redeemed with a higher redeem rate in any subsequent state Γ' which preserves the funds ratio r_1/r_0. Note that swaps are the only actions that may affect the redeem rate along the run $\Gamma \to^* \Gamma'$.

Therefore, performing swaps that eventually re-align the funds ratio to r_1/r_0 incentivizes deposits.

The condition of constant funds ratio in Theorem 5 is practically relevant. For instance, for stable exchange rates, such as in the case of exchanges between stable coins [6], the arbitrage game ensures stable fund ratios: users are hence incentivized to provide funds, as the redeem rate is likely to increase over time.

5 Related Work

To the best of our knowledge, our work is the first to study AMMs abstracting from the swap invariant. All works in literature consider concrete swap invariants; most of them focus on the constant product, popularized by Uniswap [13]. The arbitrage problem for constant-product swap invariants has been formalized in [17,19], which show that the solution can be efficiently computed, and suggest that constant product AMMs accurately tend towards exchange rates. Our work generalizes such results. Furthermore, as we have shown in §4.3 (19), the fee-rate ϕ determines how much AMMs deviate from global exchange rates: higher fees, however, also result in reduced swap amounts in arbitrage actions, negatively affecting fee accrual. In [24], the optimal fee-rate that maximizes the fee accrual for the depositing user is analytically derived.

A executable model of Uniswap [13] has been specified in [1] to analyze integer rounding errors in the Uniswap implementation.

A few alternatives to the constant product invariant have been proposed. Curve features a peculiar invariant [22] optimized for large swap volumes between *stable coins*, where the swap rate can support large amounts with small sensitivity. To efficiently compute swap invariant, implementations perform numerical approximations [15]. Should these approximations fail to converge, these implementations still guarantee that the AMM remains liquid. We conjecture that the invariants in [3, 22] are incentive-consistent. The work [25] proposes a constant product invariant that is adjusted dynamically based on the oracle price feed, thus reducing the need for arbitrage transactions, but at the cost of lower fee accrual. AMMs with *virtual* balances have been proposed [2] and implemented [7,8]. In these AMMs, the swap rate depends on past actions, besides the current funds balances in the AMM. This, similarly to [25], aims to minimize the need for arbitrage transactions to ensure the local AMM swap rate tends towards the exchange rates.

Some implementations [3] generalise AMM pairs to n-tokens, allowing users to swap any non-intersecting sets of token types. For example, the constant-product invariant becomes $I(r_0, \ldots, r_n) = r_0^{w_0} \cdot \ldots \cdot r_n^{w_n}$ where $\sum_{i=0}^{n} w^i = 1$.

6 Conclusions

We have proposed a theory of AMMs, featuring a model of their behaviour and a formally proven set of fundamental properties, characterizing both structural and economic aspects. Our theory is parametric w.r.t. platform-specific features

(e.g., swap invariants), and it abstracts from implementation-specific features, and from the features that are orthogonal to the core functionality of AMMs (e.g., governance).

There are some differences between our model and the existing AMM platforms. Uniswap implements flash-loans as part of the swap actions: namely, the user can optionally borrow available pair funds [10] whilst returning these within the same *atomic group* of actions. Further, Uniswap implements an exchange rate oracle, allowing smart contracts to interpret (averages of) recent swap rates as exchange rates [11]. Balancer [3] extends token pairs to token *tuples*: a user can swap any two non-coinciding sets of supported tokens, such that the swap invariant is maintained. In all AMM implementations, token balances are represented as integers: consequently, they are subject to rounding errors [1]. AMM platforms frequently implement a governance logic, which allow "governance token" holders to coordinate changes to AMM fee-rates or swap invariant parameters.

AMM platforms like Uniswap [13] and Curve [22] have overtaken centralized cryptocurrency markets in size and usage. On the one hand, a better understanding of AMM design in cases where AMMs host the majority of the token's global swap volume is critical [18]. It would be interesting to investigate how our theory can be used to formally explain such behaviours. On the other hand, the growth of AMMs is making them more attractive for malicious users. Current research efforts [21, 23, 28, 31] are devoted to understanding vulnerabilities and attacks, which we plan to investigate formally, exploiting our theory.

This paper, together with our work on formalizing another DeFi archetype called *lending pool* [20], is the first step towards a general theory of DeFi. We believe that a general theory encompassing interactions between different DeFi archetypes is crucial to be able to reason about their structural, economic and security aspects, as typical DeFi applications operate within a wider ecosystem, composed by a set of collaborating or competing agents, which interact through possibly separate execution environments.

Acknowledgements Massimo Bartoletti is partially supported by Conv. Fondazione di Sardegna & Atenei Sardi project F74I19000900007 *ADAM*. James Hsin-yu Chiang is supported by the PhD School of DTU Compute. Alberto Lluch Lafuente is partially supported by the EU H2020-SU-ICT-03-2018 Project No. 830929 CyberSec4Europe (cybersec4europe.eu).

References

1. Formal specification of constant product market maker model & implementation (2018), https://github.com/runtimeverification/verified-smart-contracts/blob/uniswap/uniswap/x-y-k.pdf
2. Improving frontrunning resistance of x*y=k market makers (2018), https://ethresear.ch/t/improving-front-running-resistance-of-x-y-k-market-makers/1281
3. Balancer whitepaper (2019), https://balancer.finance/whitepaper/
4. Curve statistics (2020), https://www.curve.fi/dailystats

5. Curve website (2020), https://www.curve.fi
6. Makerdao website (2020), https://https://makerdao.com
7. Mooniswap implementation (2020), https://github.com/1inch-exchange/mooniswap/blob/02dccfab2ddbb8a409400288cb13441763370350/contracts/Mooniswap.sol
8. Mooniswap whitepaper (2020), https://mooniswap.exchange/docs/MooniswapWhitePaper-v1.0.pdf
9. Uniswap Documentation: Understanding Returns (2020), https://uniswap.org/docs/v2/advanced-topics/understanding-returns/
10. Uniswap flash loan implementation (2020), https://github.com/Uniswap/uniswap-v2-core/blob/4dd59067c76dea4a0e8e4bfdda41877a6b16dedc/contracts/UniswapV2Pair.sol#L172
11. Uniswap oracle template (2020), https://github.com/Uniswap/uniswap-v2-periphery/blob/dda62473e2da448bc9cb8f4514dadda4aeede5f4/contracts/examples/ExampleOracleSimple.sol
12. Uniswap statistics (2020), https://info.uniswap.org
13. Uniswap token pair implementation (2020), https://github.com/Uniswap/uniswap-v2-core/blob/4dd59067c76dea4a0e8e4bfdda41877a6b16dedc/contracts/UniswapV2Pair.sol
14. Uniswap website (2020), https://www.uniswap.org
15. Curve computation of invariant constant (2021), https://github.com/curvefi/curve-contract/blob/a1b5a797790d3f5ef12b0e358892a0ce47c12f85/contracts/pool-templates/base/SwapTemplateBase.vy#L206
16. Curve token pair implementation (2021), https://github.com/curvefi/curve-contract/blob/a1b5a797790d3f5ef12b0e358892a0ce47c12f85/contracts/pool-templates/base/SwapTemplateBase.vy
17. Angeris, G., Chitra, T.: Improved price oracles: Constant function market makers. In: ACM Conference on Advances in Financial Technologies (AFT). pp. 80–91. ACM (2020). https://doi.org/10.1145/3419614.3423251, https://arxiv.org/abs/2003.10001
18. Angeris, G., Evans, A., Chitra, T.: When does the tail wag the dog? curvature and market making. arXiv preprint arXiv:2012.08040 (2020), https://arxiv.org/abs/2012.08040
19. Angeris, G., Kao, H.T., Chiang, R., Noyes, C., Chitra, T.: An analysis of Uniswap markets. Cryptoeconomic Systems Journal (2019), https://ssrn.com/abstract=3602203
20. Bartoletti, M., Chiang, J.H., Lluch-Lafuente, A.: SoK: Lending pools in decentralized finance. In: Workshop on Trusted Smart Contracts. LNCS, Springer (2021), (To appear)
21. Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., Breidenbach, L., Juels, A.: Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In: IEEE Symposium on Security and Privacy. pp. 910–927. IEEE (2020). https://doi.org/10.1109/SP40000.2020.00040
22. Egorov, M.: Stableswap - efficient mechanism for stablecoin (2019), https://www.curve.fi/stableswap-paper.pdf
23. Eskandari, S., Moosavi, S., Clark, J.: SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain. In: Financial Cryptography. pp. 170–189. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-43725-1_13
24. Evans, A., Angeris, G., Chitra, T.: Optimal fees for geometric mean market makers (2021), https://web.stanford.edu/~guillean/papers/g3m-optimal-fee.pdf

25. Krishnamachari, B., Feng, Q., Grippo, E.: Dynamic curves for decentralized autonomous cryptocurrency exchanges. arXiv preprint arXiv:2101.02778 (2021), https://arxiv.org/abs/2101.02778
26. Massimo Bartoletti and James Hsin-yu Chiang and Alberto Lluch-Lafuente: A theory of Automated Market Makers in DeFi. arXiv preprint arXiv:2102.11350 (2021), https://arxiv.org/abs/2102.11350
27. Mazurkiewicz, A.W.: Basic notions of trace theory. In: Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency. LNCS, vol. 354, pp. 285–363. Springer (1988). https://doi.org/10.1007/BFb0013025
28. Qin, K., Zhou, L., Gervais, A.: Quantifying blockchain extractable value: How dark is the forest? (2021), https://arxiv.org/abs/2101.05511
29. Qin, K., Zhou, L., Livshits, B., Gervais: Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit. In: Financial Cryptography (2021), (to appear) https://arxiv.org/abs/2003.03810
30. Wang, D., Wu, S., Lin, Z., Wu, L., Yuan, X., Zhou, Y., Wang, H., Ren, K.: Towards understanding flash loan and its applications in defi ecosystem. arXiv preprint arXiv:2010.12252 (2020), https://arxiv.org/abs/2010.12252
31. Zhou, L., Qin, K., Torres, C.F., Le, D.V., Gervais, A.: High-Frequency Trading on Decentralized On-Chain Exchanges. arXiv preprint arXiv:2009.14021 (2020), https://arxiv.org/abs/2009.14021