DTU Compute
Department of Applied Mathematics and Computer Science

A Formal Approach to AMM Fee Mechanisms with Lean 4

Master Thesis

A Formal Approach to AMM Fee Mechanisms with Lean 4

Master Thesis
June, 2025

By
Marco Dessalvi

Supervised By
Alberto Lluch-Lafuente, Technical University of Denmark
Massimo Bartoletti, University of Cagliari

Cover photo: Vibeke Hempler, 2012
Published by: DTU, Department of Applied Mathematics and Computer Science, Richard Petersens Plads, Building 324, 2800 Kgs. Lyngby Denmark
www.compute.dtu.dk

Abstract

Decentralized Finance (DeFi) has revolutionized financial market by enabling automated token exchanges without a central authority as an intermediary. Pivotal to many DeFi platforms are Automated Market Makers (AMMs). There are a number of AMMs implementations, many of which use the constant product swap rate. In the real world, AMMs impose a trading fee that affects liquidity providers' incentives and users' arbitrage strategies. In this work, we extend some foundational models, introducing a trading fee ϕ into the swap rate formula, which reduces to the foundational model when $\phi = 1$. We analyze the new model from an economic point of view, proving that the fee-adjusted constant product swap rate keeps some of the key properties that it had in the foundational model (output-boundness, monotonicity), and we also prove that it does not satisfy some of the old ones, mainly additivity, which is instead generalized to explain the behavior of the function in the presence of a trading fee. In particular, we show that swapping larger amounts of tokens yields strictly greater profit than splitting trades when $\phi < 1$. Finally, we analyze and provide a solution to the arbitrage problem, deriving the optimal swap amount and the amount that aligns the price induced by the AMM to the actual market price given by an external oracle, revealing that they both diverge from the no-fee model.

Acknowledgements

First I would like to thank my two supervisors, Alberto and Massimo, who have helped and guided me throughout the writing of this work. I want to deeply thank my family, who have always stood by and supported me, even in these two years of distance. I also want to thank all of the friends that I met during this journey, who always bring a smile to my face and have become like a second family here. Lastly, I want to thank all of my friends back home, who always make it feel like home whenever I come back.

Contents

Chapter 1

Introduction

Decentralized Finance (DeFi) has transformed the financial landscape by enabling peer-to-peer transactions without a central intermediary. One of the core mechanisms of many DeFi applications is the Automated Market Makers (AMMs), which enable token swaps through algorithmic pricing mechanisms. Among the various kinds of AMMs, the Constant Product Market Makers (CPMMs) are the most used ones, with implementations such as Uniswap v2/v3 [1] [2], Sushiswap [3], *et cetera*. To better understand the importance of these implementations, as of May 2025, Uniswap stands out with an approximate Total Value Locked (TVL) of $5 billion [4].

While AMMs have been extensively studied, the role of trading fees - which is critical in the real-world implementations of AMMs - has often gone under-looked in formal analyses of AMMs models. Trading fees play a crucial role on incentivizing liquidity provision, but also on influencing user strategies and the overall market dynamics. A formal approach and understanding of how fees impact AMM behavior is essential for both theoretical and practical applications.

This thesis aims to extend the already existing models [5] and formalization [5] addressing this gap by providing a formal analysis of AMMs incorporating trading fees. With the use of Lean 4 [6] we extend the existing formalization of AMMs to include the fee mechanisms for token swaps. This work builds on foundational formalizations, enriching them to model economic properties introduced — or altered — by trading fees.

In the following sections of this work, we present numerous definitions and lemmas, each accompanied by their respective implementations in Lean 4. To help the reader navigate between the theoretical content of this work and the practical implementation, we use integrated interactive buttons [7] adjacent to each definition and lemma. Clicking these buttons redirects the reader to the corresponding entries in the project's documentation, providing access to the formal Lean 4 implementation. Under each lemma, there is also a pen-and-paper proof outline that outlines the proof strategy used in Lean to prove the respective lemma. For a full overview of the documentation, the docs index can be accessed by clicking the button on the left or at https://mamboleano.github.io/lean4-amm-fees/.

1.1 Related Work

The literature is fairly rich in works that study AMMs. In this section, we mainly focus on the foundational works on which this thesis is based, and some others that analyze the fee mechanisms in AMMs.

The foundational work [8] presents an abstract operational model on which the implementation of this thesis is based. This model does not depend on any specific implementation, but it simply identifies the conditions necessary to ensure desirable properties from AMMs, like net worth preservation and arbitrage conditions. This is the theoretical framework that describes the AMM design that this thesis works with. In addition to [8], this thesis delves into the formal verification of AMMs incorporating trading fees, while the model in [8] does not take these fees into account, but proposes this as possible future work. While it is true that all the structural properties presented in [8] still hold after introducing the trading fee in the model (because they do not depend on it), some of the other key properties don't:

1. **Additivity**: Given two consecutive swap rates

$$\alpha = SX_\phi(x, r_0, r_1) \text{ and } \beta = SX_\phi(y, r_0 + x, r_1 - \alpha x)$$

   where r_0 and r_1 are the initial reserves of the AMM pair, according to the definition given in [8], the swap rate SX_ϕ is additive if:

$$SX_\phi(x + y, r_0, r_1) = \frac{\alpha x + \beta y}{x + y}$$

   In other terms, swapping an amount of tokens $x + y$ should be equal to swapping first an amount x and subsequently an amount y. However, the fee-adjusted constant product swap rate does not fit the definition of additivity from [8]. Instead, we give our generalized definition that also takes into account the trading fee by multiplying the previous formula with the factor $\zeta_\phi(x, y, r_0, r_1)$:

$$SX_\phi(x + y, r_0, r_1) = \frac{\alpha x + \beta y}{x + y} \cdot \zeta_\phi(x, y, r_0, r_1)$$

2. **Reversibility**: Given a swap rate $\alpha = SX_\phi(x, r_0, r_1)$ where r_0 and r_1 are the initial reserves of the AMM pair, the work [8] says that SX_ϕ is reversible if:

$$SX_\phi(\alpha x, r_1 - \alpha x, r_0 + x) = \frac{1}{\alpha}$$

   In other terms, if after performing a swap the trader swaps again all of the output tokens that he just received, the AMM would go back to the original state.

   The fee-adjusted constant product swap rate, however, is not reversible. The product of the AMM's reserves strictly increases after each swap, so there is no swap that can bring the reserves back to the original state. Hence, the proof strategies in this work do not make use of this definition.

3. **Equilibrium vs Arbitrage**: The equilibrium value is the value that a user A can swap to bring the price induced by the AMM equal to the one induced by the external oracle. In [8], this value coincides with the arbitrage solution, i.e. it is also the value that maximizes A's profit. This, however, is not the case in the fee-bearing model, and we show that the equilibrium value is different from the optimal one.

Another foundational work is the Lean 4 implementation of such model provided in [5]. This work models the core functionalities of AMMs, including liquidity provision and token swaps. It also provides machine-checked proofs of the relevant economic properties presented, with the main result being the arbitrage solution, ensuring its mathematical rigor. This thesis extends this work, adding an implementation for trading fees and providing the

appropriate formal definitions in the Lean 4 theorem prover. Unlike [5], this thesis does not take into account liquidity provision, which means that is solely focuses on token swaps, so it could be interesting to analyze how their incentives change with the introduction of fees in future work. Like in the former implementation, ours provides full machine-checked proofs of all the results provided, with the main one being the arbitrage solution, which differs from the one presented in [8] due to the fees mechanisms. While in the previous work this solution coincided with the value that brought the internal swap rate of the AMM to be aligned with the external prices of the AMM tokens, in our model it is not the case.

The work [9] considers two-assets Constant Functions Market Makers (CFMMs [10]), while in our work we focus more on Constant Product Market Makers (CPMMs), a subset of those. The model in [9] is also very different from ours: it is based on a previous work described in [11], and the fees are modeled in the context of discrete Poisson block generation. Hence, it also presents very different results from our work, focusing more on the impact of fees on the arbitrageurs over time. Also, to the best of our knowledge, the results presented in [9] are not backed up by any machine-checked formalization, but only present pen-and-paper proofs.

There are also works that analyze different aspects of fees. For example, the work [12] focuses more on the protocol fee. The protocol fee is a portion of the trading fee that is diverted to the protocol, mainly used to create some revenue for the governance token holders . The work [12] also analyzes the different incentives from both the liquidity providers and the traders. The model which they use is also quite different from ours, as it models features like sticky liquidity and trade volume.

In summary, our research advances this previous work by incorporating trading fees into the theoretical model and into the implementation, bringing both closer to the real-world scenario and contributing to the development of more secure and robust DeFi systems.

1.2 Contributions

We summarize our contributions as follows:

1. **Formal Modeling**: We develop a formal model of AMMs that integrates trading fees, capturing their effect on swap outcomes and user's incentives. We introduce this trading fee in the constant product swap rate formula (Definition 2.4.1) and discuss the similarities between this model and the implementation of the swap in Uniswap v2.

2. **Economic Analysis**: We analyze core economic properties of AMMs incorporated with fees, such as how they impact user's gain on token swaps, arbitrage opportunities and exchange rates. In addition to the properties from the foundational work [8], we provide a more general definition of *additivity* (Definition 3.2.7), that takes into account the trading fee, analyzing the consequences that implies on a user's gain in Theorem 3.2.8. We also study the relationship between the internal rate, the swap rate and the external exchange rate in Lemmas 3.4.1 and 3.4.2, which is crucial to understand and solve the arbitrage problem. Finally, we provide a concrete value that brings the AMM to equilibrium in Theorem 3.5.1, analyzing the gain that a user A would get by swapping this amount of tokens in Theorem 3.5.4, and we provide a solution to the arbitrage problem, defining an optimal value in Theorem 3.5.5.

3. **Lean 4 Formalization**: The major contribution of this work. We provide a complete formalization of all the new definitions, lemmas and theorems introduced in this thesis as well as their formal proof, extending the library presented in [5]. This

ensures mathematical rigor and serves as a formal verification on all the properties presented.

Providing a formal verification on the role of trading fees in AMMs, this thesis contributes to the ongoing efforts to improve the robustness and security of DeFi systems.

Chapter 2

AMM model

In this chapter, we present the formal operational model of AMMs described in [8] with a few simplifications. Since our focus is mainly on swap transactions, we set aside any model details that don't impact (or are not affected by) swap actions.

Compared to the original model, we introduce the trading fees in the swap rate function, which brings this model closer to the Uniswap v2 protocol [1]. The structure and content of the following sections largely mirrors that of [8], with the exception of Section 2.4,where we introduce the trading fee in the model and contrast it with Uniswap v2's actual implementation.

2.1 Tokens and Wallets

The model presented in [8] assumes a set $\mathbb{T}_0$ of ***atomic token types***, which represent native cryptocurrencies and application-specific tokens. For instance, $\mathbb{T}_0$ may include ETH, the native cryptocurrency of Ethereum, and WBTC, i.e. Bitcoins wrapped with the ERC20 interface for Ethereum tokens.

The model also assumes a set $\mathbb{T}_1$ of ***minted token type***, which is an unordered pair of distinct atomic token types: assuming that τ_0 and τ_1 are atomic token types and that $\tau_0 \neq \tau_1$, then $\{\tau_0, \tau_1\}$ is a valid minted token type which represents shares in the AMM holding the reserves of τ_0 and τ_1. Also, since it is an unordered pair, $\{\tau_0, \tau_1\}$ and $\{\tau_1, \tau_0\}$ represent the same minted token type. Since our work focuses mainly in swap transactions and not so much on liquidity provision, the use of minted token types is this thesis is very limited.

In this model, tokens are *fungible*, i.e. individual units of the same type are interchangeable. This means that 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. This is particularly useful for our proofs later on in this work.

We write $\mathbb{T}$ for the universe of all token types, i.e. $\mathbb{T} = \mathbb{T}_0 \cup \mathbb{T}_1$, and we use $\tau, \tau', \ldots$ to range over $\mathbb{T}$. We write $r : \tau$ to denote r units of a token of type τ, either atomic or minted.

The model also introduces a set of ***users*** $\mathbb{A}$, ranged over by $\mathsf{A}, \mathsf{A}', \mathsf{B} \ldots$ We model the ***wallet*** of a user A as a term $\mathsf{A}[\sigma_\mathsf{A}]$, where the finite partial map $\sigma_\mathsf{A} \in \mathbb{T} \rightharpoonup \mathbb{R}_{\geq 0}$ represents A's token balance. Note how the balance of any user is strictly non-negative, for both atomic and minted token types.

2.2 States

In [8], an ***AMM*** holding r_0 units of τ_0 and r_1 units of τ_1 is defined as an unordered pair $\{r_0 : \tau_0, r_1 : \tau_1\}$. We stress again that in order to be a valid AMM, we require $\tau_0 \neq \tau_1$ and $r_0, r_1 \in \mathbb{R}_{>0}$. Also, since it is an unordered pair, $\{r_0 : \tau_0, r_1 : \tau_1\}$ and $\{r_1 : \tau_1, r_0 : \tau_0\}$ define the same AMM.

The interaction between users and AMMs is modelled as a labelled transition system (LTS). Its labels represent blockchain ***transactions***, while the ***states*** $\Gamma, \Gamma', \Delta, \ldots$ are finite non-empty compositions of wallets and AMMs. Formally, states are terms of the form:

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

and subject to the following conditions. For all $i \neq j$:

1. $\mathsf{A}_i \neq \mathsf{A}_j$ (each user has a single wallet);
2. $\{\tau_i, \tau_i'\} \neq \{\tau_j, \tau_j'\}$ (distinct AMMs cannot hold exactly the same token types).

Note that these conditions allow AMMs to have a common token type τ, e.g. 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. We stipulate that the ordering of terms in a state is immaterial. Hence, we consider two states Γ and Γ' to be equivalent when they contain the same terms (regardless of their order). For a term Q and a state Γ, we write $Q \in \Gamma$ when $\Gamma = Q \mid \Gamma'$, for some Γ'. Compared to [8] we do not focus on *initial* or *reachable* states in this work, since we only deal with swap transactions.

In some case, it is useful to refer to the total amount of tokens τ in a state Γ. This amount is defined as $S_\Gamma \tau$ that denotes the ***supply*** of τ in Γ. This term is defined as the sum of the reserves of τ in all the wallets and the AMMs in Γ. More formally:

$$S_{\mathsf{A}[\sigma_\mathsf{A}]}\tau = \begin{cases} \sigma_\mathsf{A}\tau & \text{if } \tau \in \operatorname{dom}\sigma_\mathsf{A} \\ 0 & \text{otherwise} \end{cases} \qquad S_{\{r_0:\tau_0, r_1:\tau_1\}}\tau = \begin{cases} r_i & \text{if } \tau = \tau_i \\ 0 & \text{otherwise} \end{cases} \qquad S_{\Gamma \mid \Gamma'}\tau = S_\Gamma\tau + S_{\Gamma'}\tau$$

Note that this is defined over induction of the structure of states. In the first case we are considering the reserves of τ is a user's A wallet. If τ is present in the domain of σ_A, then the value is retrieved from the mapping (i.e. the number of tokens τ held by A). Otherwise, it means that A does not hold any reserve of τ, thus 0 is added to the total supply. In the second case, we are considering the reserves of τ in an AMM $\{r_0 : \tau_0, r_1 : \tau_1\}$. If $\tau = \tau_0$, then we add r_0 to the total supply; if $\tau = \tau_1$, then we add r_1 to the total supply, otherwise, we add 0.

> **Example 2.2.1.** Let $\Gamma = \mathsf{A}[2 : \tau_0, 3 : \tau_1, 3 : \{\tau_0, \tau_1\}] \mid \mathsf{B}[3 : \tau_0, 6 : \tau_2] \mid \{5 : \tau_0, 6 : \tau_1\}$. Then we have that $S_\Gamma\tau_0 = 2+3+5 = 10$, $S_\Gamma\tau_1 = 3+6 = 9$, $S_\Gamma\tau_2 = 6$ and $S_\Gamma\{\tau_0, \tau_1\} = 3$

2.3 Transactions

State transitions are triggered by transactions $\mathsf{T}, \mathsf{T}', \ldots$, which can have the following form (where τ_0 and τ_1 are atomic tokens):

$$\mathsf{A} : \mathsf{swap}(v, \tau_0, \tau_1)$$

With this transaction, A transfers $v : \tau_0$ to an AMM $\{r_0 : \tau_0, r_1 : \tau_1\}$, receiving in return some units of τ_1, which are removed from the AMM.

In the rest of this work, we only deal with swap transactions. Any user A can swap v units of τ_0 in her wallet for some units of τ_1 in an AMM $\{r_0 : \tau_0, r_1 : \tau_1\}$ through the transaction $\mathsf{A} : \mathsf{swap}(v, \tau_0, \tau_1)$. Symmetrically, A can swap v of her units of τ_1 for units of τ_0 in the AMM through a transaction $\mathsf{A} : \mathsf{swap}(v, \tau_1, \tau_0)$. The ***swap rate*** $SX_\phi(x, r_0, r_1)$ determines the amount of *output tokens* τ_1 that a user receives upon an amount of x *input tokens* τ_0 in an AMM $\{r_0 : \tau_0, r_1 : \tau_1\}$. Below is the formal semantic rule for the swap transaction.

$$\frac{\sigma\tau_0 \geq x \qquad y = x \cdot SX_\phi(x, r_0, r_1) < r_1}{\mathsf{A}[\sigma] \mid \{r_0 : \tau_0, r_1 : \tau_1\} \mid \Gamma \xrightarrow{\mathsf{A}:\mathsf{swap}(x,\tau_0,\tau_1)} \mathsf{A}[\sigma - x : \tau_0 + y : \tau_1] \mid \{r_0 + x : \tau_0, r_1 - y : \tau_1\} \mid \Gamma} \text{[SWAP]}$$

To sum up, we say that a swap transaction $\mathsf{A} : \mathsf{swap}(v, \tau_1, \tau_0)$ is ***enabled*** in a blockchain state Γ if A has enough tokens in his wallet (i.e. $\sigma_\mathsf{A}\tau_0 \geq x$), if the AMM $\{r_0 + x : \tau_0, r_1 - y : \tau_1\}$ exists in Γ and if the reserves of such AMM would not get drained after the swap (i.e. $x \cdot SX_\phi(x, r_0, r_1) < r_1$). Hence, given a transition $\mathsf{T}(x) = \mathsf{A} : \mathsf{swap}(x, \tau_1, \tau_0)$, assuming that $\mathsf{T}(x)$ is enabled in Γ, or equivalently writing that $\mathsf{T}(x)$ is s.t. $\Gamma \xrightarrow{\mathsf{T}(x)} \Gamma'$ means assuming the three conditions stated above.

2.4 Constant product swap rate

We can see that the swap action requires a swap rate function SX_ϕ. In Uniswap v2 [13] and in many other mainstream AMM implementations [14, 3], the chosen swap rate function is the ***constant product swap rate***.

Definition 2.4.1 ▶ Constant product swap rate

The constant product swap rate function is:

$$SX_\phi(x, r_0, r_1) = \frac{\phi\, r_1}{r_0 + \phi\, x} \qquad \text{where } \phi \in (0, 1]$$

This function involves a trading fee ϕ. When no fee is applied (i.e. when $\phi = 1$), the constant product swap rate is the same as the one presented in [8]. A fee that is strictly less than 1 results in a reduced amount of output tokens received from the actors that perform the swap. But how exactly does this work and why does it benefit liquidity providers? The intuition is that when the transaction $\mathsf{A} : \mathsf{swap}(v, \tau_0, \tau_1)$ is fired, the effective number of tokens that are used to re-balance the AMM pool is not v, but $\phi \cdot v$ instead. Now, this means that A receives fewer tokens and some units of τ_1 token are "kept" by the AMM in order to benefit the liquidity providers (LPs).

Example 2.4.2. Let $\Gamma = \mathsf{A}[30 : \tau_0,\ 20 : \tau_1] \mid \{40 : \tau_0, 60 : \tau_1\}$. Now suppose that A wants to swap 10 units of τ_0. Assume that we are instantiating SX_ϕ with the constant product swap rate function defined in 2.4.1, with $\phi = 0.997$. Then, A expects to receive:

$$y = 10 \cdot \frac{0.997 * 60}{40 + 0.997 \cdot 10} \approx 11.97$$

τ_1 tokens. Hence, after firing $\mathsf{T} = \mathsf{A} : \mathsf{swap}(10, \tau_0, \tau_1)$, i.e. $\Gamma \xrightarrow{\mathsf{T}} \Gamma'$, the resulting state is $\Gamma' = \mathsf{A}[20 : \tau_0,\ 31.97 : \tau_1] \mid \{50 : \tau_0, 48.03 : \tau_1\}$. Now, to back up our previous claims we also show the amount of τ_1 units A would expect if the AMM used the constant product

swap rate used in [8], i.e. when $\phi = 1$. With this, A would expect:

$$y = 10 \cdot \frac{1 * 60}{40 + 1 \cdot 10} = 10 \cdot \frac{60}{40 + 10} = 12$$

The resulting state would be $\Gamma'' = \mathsf{A}[20 : \tau_0,\ 32.00 : \tau_1] \mid \{50 : \tau_0, 48.00 : \tau_1\}$. As we can see, in both the resulting states Γ' and Γ'' the AMM has 50 units of τ_0 tokens. However, in Γ', the AMM retains a small portion of the τ_1 tokens compared to Γ'', which results in a slightly worse outcome for the trader A and a better one for the LPs.

The name of the function is actually misleading: the product of the two reserves of the AMM $r_0 \cdot r_1$ is constant only when there is no fee involved (as in the model presented in [8]). However, when the fee is introduced, the product does not remain constant, but it increases after each swap. So, if an AMM $\{r_0 : \tau_0, r_1 : \tau_1\}$ evolves into $\{r_0 + x : \tau_0, r_1 - y : \tau_1\}$ upon a swap, then the constant product swap rate ensures that:

$$(r_0 + x)(r_1 - y) \ \geq \ r_0 r_1 \tag{2.1}$$

Proof of (2.1). We develop the left side of the inequality first:

$$\begin{aligned}
&(r_0 + x)(r_1 - y) \\
=&(r_0 + x)\left(r_1 - \frac{\phi x r_1}{r_0 + \phi x}\right) \\
=&r_1(r_0 + x)\left(1 - \frac{\phi x}{r_0 + \phi x}\right) \\
=&r_1(r_0 + x)(\frac{r_0 + \phi x - \phi x}{r_0 + \phi x}) \\
=&r_1(r_0 + x)(\frac{r_0}{r_0 + \phi x}) \\
=&r_0 r_1(\frac{r_0 + x}{r_0 + \phi x})
\end{aligned}$$

So the inequality becomes:

$$\begin{aligned}
& r_0 r_1(\frac{r_0 + x}{r_0 + \phi x}) \geq r_0 r_1 \\
\iff & (\frac{r_0 + x}{r_0 + \phi x}) \geq 1 \\
\iff & r_0 + x \geq r_0 + \phi x \\
\iff & x \geq \phi x \\
\iff & 1 \geq \phi
\end{aligned}$$

And this is true by assumption. □

This condition is also specified in Uniswap v2's implementation [1] that can also be seen in the Listing 4.5 at page 44.

[1] https://github.com/Uniswap/v2-core/blob/master/contracts/UniswapV2Pair.sol

2.5 Prices and Exchange Rate

To model the prices of atomic tokens, the model from [8] introduces an external oracle that assigns a price to each atomic token $\tau \in \mathbb{T}_0$. In particular, this external oracle is modeled as a function $P : \mathbb{T}_0 \to \mathbb{R}_{>0}$. With this, the model assumes that the prices of atomic tokens stay constant during an execution. However, prices of minted tokens depend on the state of the AMM. More precisely, it depends both on the supply of the minted token in the users' wallets and on the reserves of the respective atomic tokens in the AMM. Formally, the price of a minted token $\{\tau_0, \tau_1\}$ in a state Γ is defined as:

$$P_\Gamma\{\tau_0, \tau_1\} = \frac{r_0 \cdot P\tau_0 + r_1 \cdot P\tau_1}{S_\Gamma\{\tau_0, \tau_1\}} \qquad \text{if } \{r_0 : \tau_0, r_1 : \tau_1\} \in \Gamma$$

For uniformity, we say that $P_\Gamma\tau = P\tau$ when $\tau \in \mathbb{T}_0$. The idea behind the above equation is that the value of one minted token must be equal to the value of the atomic tokens that can be obtained by redeeming the minted token. Since in this work we don't consider any redeem transaction, we overlook this. However, all the considerations and structural properties described in [8] still hold, as they do not depend on any particular assumption about the swap rate function.

A key question at this point should be, how do we determine the exchange rate between two atomic token types τ_0 an τ_1? This ***exchange rate*** $X(\tau_0, \tau_1)$ is dictated by the prices given by the external oracle:

$$X(\tau_0, \tau_1) = \frac{P\tau_0}{P\tau_1} \tag{2.2}$$

This defines how many units of τ_1 tokens a user can buy with 1 unit of τ_0. This only depends on the prices given by the external oracle, completely ignoring the state of the AMM. However, AMMs also induce an exchange rate based on the current state of the AMM's reserves and the swap rate function that is being used. More precisely, the ***internal exchange rate*** $X_\Gamma(\tau_0, \tau_1)$ between two atomic token types τ_0 and τ_1 in a state Γ is the limit of the swap rate function SX_ϕ as x approaches 0:

$$X_\Gamma(\tau_0, \tau_1) = \lim_{x \to 0} SX_\phi(x, r_0, r_1) \qquad \text{if } \{r_0 : \tau_0, r_1 : \tau_1\} \in \Gamma \tag{2.3}$$

The intuition behind this equation is similar to the previous one: for a very small x, a user swapping x units of τ_0 tokens would expect to receive $x \cdot X_\Gamma(\tau_0, \tau_1)$ units of τ_1 tokens. We will see later in this work that rational users try to align this internal exchange rate with the one given by the external oracle.

> **Example 2.5.1.** Let $\Gamma = \mathsf{A}[30 : \tau_0,\ 20 : \tau_1] \mid \{40 : \tau_0, 60 : \tau_1\}$. Now suppose that the prices of τ_0 and τ_1 are $P(\tau_0) = 4$ and $P(\tau_1) = 5$. Hence, the exchange rate is:
>
> $$X(\{\tau_0, \tau_1\}) = \frac{4}{5}$$
>
> Assuming that we are using the constant product swap rate, and that $\phi = 0.997$, the internal exchange rate computed in the state Γ is:
>
> $$\begin{aligned} X_\Gamma(\tau_0, \tau_1) &= \lim_{x \to 0} SX_\phi(x, r_0, r_1) \\ &= \lim_{x \to 0} SX_\phi(x, 40, 60) \end{aligned}$$

$$
\begin{aligned}
&= \lim_{x \to 0} \frac{0.997 \cdot 60}{40 + 0.997 \cdot x} \\
&= \frac{0.997 \cdot 60}{40} \\
&= 1.4955
\end{aligned}
$$

In the next example, we show that a trader is incentivized to align these two exchange rates.

2.6 Wealth measures

Since one of the primary goals of this paper is to provide a solution to the arbitrage problem, we also have define how to measure a user's wealth, in order to determine when it increases. The model presented in [8] defines the ***net worth*** of a user A as the measure of A's wealth in tokens (both atomic and minted). Formally, the net worth of A in a state Γ is defined as:

$$
W_{\mathsf{A}}(\Gamma) = \begin{cases} \sum_{\tau \in \operatorname{dom} \sigma_{\mathsf{A}}} \sigma_{\mathsf{A}} \tau \cdot P_{\Gamma}(\tau) & \text{if } \mathsf{A}[\sigma_{\mathsf{A}}] \in \Gamma \\ 0 & \text{otherwise} \end{cases} \tag{2.4}
$$

To determine when a user is incentivized to perform a swap, the model also defines the the ***gain*** of user A upon performing a transactions T enabled in state Γ (if T is not enabled in Γ, then the gain is zero) as follows:

$$
G_{\mathsf{A}}(\Gamma, \mathsf{T}) = W_{\mathsf{A}}(\Gamma') - W_{\mathsf{A}}(\Gamma) \qquad \text{if } \Gamma \xrightarrow{\mathsf{T}} \Gamma' \tag{2.5}
$$

Example 2.6.1. Let $\Gamma = \mathsf{A}[30 : \tau_0,\ 20 : \tau_1] \mid \{40 : \tau_0, 60 : \tau_1\}$. Now suppose that the prices of τ_0 and τ_1 are $P(\tau_0) = 4$ and $P(\tau_1) = 5$. Hence, the net worth of A in the state Γ is $W_{\mathsf{A}}(\Gamma) = 30 \cdot 4 + 20 \cdot 5 = 220$.

Now suppose that A performs the same swap as in Example 2.4.2, i.e. $\mathsf{T} = \mathsf{A} : \mathsf{swap}(10, \tau_0, \tau_1)$, that SX_ϕ is the constant product swap rate and that $\phi = 0.997$. Hence, the resulting state is $\Gamma' = \mathsf{A}[20 : \tau_0, 31.97 : \tau_1] \mid \{50 : \tau_0, 48.03 : \tau_1\}$ (note that the reserves are slightly approximated). At this point, the net worth of A is $W_{\mathsf{A}}(\Gamma') = 20 \cdot 4 + 31.97 \cdot 5 = 239.85$.

Finally, we can see that computing the gain upon firing T gives us:

$$
\begin{aligned}
G_{\mathsf{A}}(\Gamma, \mathsf{T}) &= W_{\mathsf{A}}(\Gamma') - W_{\mathsf{A}}(\Gamma) \\
&= 239.85 - 220 \\
&= 19.85
\end{aligned}
$$

So we can see that the gain is positive. Also, computing the exchange rate $X(\tau_0, \tau_1) = 0.8$ and both the internal exchange rates $X_{\Gamma}(\tau_0, \tau_1) = 1.4955$ and $X_{\Gamma'}(\tau_0, \tau_1) \approx 0.958$ we can also see that the internal exchange rate got "closer" to the exchange rate. This is one of the intuitions to solve the arbitrage problem and to understand when the trader's gain is positive.

Chapter 3

Economic properties of AMMs with trading fees

In this chapter we study the economic properties of Automated Market Makers (AMMs) that apply a trading fee on swap transactions. Our goal is to analyze how trading fees play a role on the incentives of users and the behavior and properties of swap functions, with keen focus on the constant product swap rate function (2.4.1).

All the results of this paper are formalized in Lean 4 and are based on an extension of the model introduced by Bartoletti et al. in [8], and the formalization in Lean 4 introduced by Pusceddu et al. in [5].

To help the reader understand better the importance of each lemma and result, we introduce the economic intuition behind the results, their formal statement, and - when appropriate - an illustrative example.

3.1 Gain of swap actions

One of the key properties of our AMMs model is the user's gain. As we previously stated, we only focus on the swap transactions; hence, it is only natural to measure the user's gain after a swap and to ask ourselves when this gain is positive. In the following lemma, we define the user's gain in terms of a valid swap transaction.

Lemma 3.1.1 ▶ Swap gain

Let $\Gamma = \mathsf{A}[\sigma_{\mathsf{A}}] \mid \{r_0 : \tau_0, r_1 : \tau_1\} \mid \Delta$, and let $\mathsf{T} = \mathsf{A} : \mathsf{swap}(x, \tau_0, \tau_1)$ be enabled in Γ. Then:

$$G_{\mathsf{A}}(\Gamma, \mathsf{T}) = \quad x \cdot \left(SX_{\phi}(x, r_0, r_1)\, P(\tau_1) - P(\tau_0)\right) \cdot \left(1 - \frac{\sigma_{\mathsf{A}}\{\tau_0, \tau_1\}}{S_{\Gamma}\{\tau_0, \tau_1\}}\right)$$

Proof. Proof is the same as in [8]. □

This result is not different from the one formalized in [8], simply because it does not depend on any particular swap function SX_{ϕ}. It also captures how the gain depends on the number of minted tokens held by the user $\sigma_{\mathsf{A}}\{\tau_0, \tau_1\}$ and the total supply of such minted token $S_{\Gamma}\{\tau_0, \tau_1\}$. Note that, since the [SWAP] rule does not alter these two amounts, the term $\left(1 - \frac{\sigma_{\mathsf{A}}\{\tau_0, \tau_1\}}{S_{\Gamma}\{\tau_0, \tau_1\}}\right)$ remains the same even after performing a swap. We use this later in

our proof outlines. Since we mostly focus on users that are just swapping tokens (and are not Liquidity Providers), it's worth analyzing the case where A does not hold any minted token.

Lemma 3.1.2 ▶ Swap gain no mint

Let $\Gamma = \mathsf{A}[\sigma_{\mathsf{A}}] \mid \{r_0 : \tau_0, r_1 : \tau_1\} \mid \Delta$ be such that $\sigma_{\mathsf{A}}\{\tau_0, \tau_1\} = 0$, and let $\mathsf{T} = \mathsf{A} : \mathsf{swap}(x, \tau_0, \tau_1)$ be enabled in Γ. Then:

$$G_{\mathsf{A}}(\Gamma, \mathsf{T}) = \quad x \cdot \left(SX_\phi(x, r_0, r_1)\, P(\tau_1) - P(\tau_0)\right)$$

Proof. Proof is the same as in [8]. □

Example 3.1.3. Let $\Gamma = \mathsf{A}[30 : \tau_0,\ 20 : \tau_1] \mid \{40 : \tau_0, 60 : \tau_1\}$. Now suppose that the prices of τ_0 and τ_1 are $P(\tau_0) = 4$ and $P(\tau_1) = 5$. These are the same values from Example 2.6.1.

Now let $\mathsf{T} = \mathsf{A} : \mathsf{swap}(10, \tau_0, \tau_1)$ and suppose that SX_ϕ is the constant product swap rate and that $\phi = 0.997$. From the computations made in the same example, we know that

$$G_{\mathsf{A}}(\Gamma, \mathsf{T}) = 19.85 \tag{3.1}$$

Let's see now if the formula from Lemma 3.1.2 yields the same result. For now, let's suppose that T is enabled in Γ and we instantiate SX_ϕ with the constant product swap rate. We later show that since A has enough tokens and the AMM for the pair $\{\tau_0, \tau_1\}$ already exists in Γ, then it only suffices SX_ϕ to be output-bounded in order to ensure that the AMM is not drained after performing the swap. We later prove that the constant product swap rate is output bounded in Lemma 3.3.1. With these assumptions, we can compute the gain based on Lemma 3.1.2:

$$\begin{aligned}
G_{\mathsf{A}}(\Gamma, \mathsf{T}) &= x \cdot (SX_\phi(x, r_0, r_1) P(\tau_1) - P(\tau_0)) \\
&= 10 \cdot \left(\frac{\phi r_1}{r_0 + \phi x} \cdot 5 - 4\right) \\
&= 10 \cdot \left(\frac{0.997 \cdot 60}{40 + 0.997 \cdot 10} \cdot 5 - 4\right) \\
&= 10 \cdot (1.197 \cdot 5 - 4) \\
&= 19.85
\end{aligned}$$

With this result, we can finally answer one of the key questions we have posed ourselves: when the user's gain is positive upon a valid swap transaction. The following lemma defines this in a more general way, stating that the gain is strictly positive *iff* the swap rate is strictly greater than the external exchange rate.

Lemma 3.1.4 ▶ Swap rate *vs.* exchange rate

Let $\Gamma = \mathsf{A}[\sigma_{\mathsf{A}}] \mid \{r_0 : \tau_0, r_1 : \tau_1\} \mid \Delta$ be such that $S_\Gamma\{\tau_0, \tau_1\} > 0$ and $\sigma_{\mathsf{A}}\{\tau_0, \tau_1\} < S_\Gamma\{\tau_0, \tau_1\}$, and let $\mathsf{T} = \mathsf{A} : \mathsf{swap}(x, \tau_0, \tau_1)$ be enabled in Γ. Then:

$$G_{\mathsf{A}}(\Gamma, \mathsf{T}) \circ 0 \iff SX_\phi(x, r_0, r_1) \circ X(\tau_0, \tau_1) \qquad \text{for } \circ \in \{<, =, >\}$$

Proof. Let $y = x \cdot SX_\phi(x, r_0, r_1)$ and let $\eta = 1 - \frac{\sigma_{\mathsf{A}}\{\tau_0,\tau_1\}}{S_\Gamma\{\tau_0,\tau_1\}}$. From the assumptions, note that $\eta > 0$. By Lemma 3.1.1 we have that:

$$\begin{aligned} G_{\mathsf{A}}(\Gamma, \mathsf{T}) \circ 0 &\iff \eta\,(y\,P(\tau_1) - x\,P(\tau_0)) \circ 0 \\ &\iff y\,P(\tau_1) - x\,P(\tau_0) \circ 0 \\ &\iff \frac{y}{x} \circ \frac{P(\tau_0)}{P(\tau_1)} \\ &\iff SX_\phi(x, r_0, r_1) \circ X(\tau_0, \tau_1) \end{aligned}$$

□

This lemma is the first step towards finding the arbitrage solution, since it defines when it is convenient for a user to perform a swap transaction, comparing the swap rate with the external prices of the swapped tokens τ_0 and τ_1.

Example 3.1.5. Let $\Gamma = \mathsf{A}[30 : \tau_0,\ 20 : \tau_1] \mid \{60 : \tau_0, 40 : \tau_1\}$. Now suppose that the prices of τ_0 and τ_1 are $P(\tau_0) = 4$ and $P(\tau_1) = 5$. Note that these are the same values as the previous examples with the only exception of the AMM's reserves that are inverted. Suppose that SX_ϕ is the constant product swap rate and that $\phi = 0.997$. Then,

$$SX_\phi(10, r_0, r_1) = \frac{0.997 \cdot 40}{60 + 0.997 \cdot 10} = 0.57$$
$$X(\tau_0, \tau_1) = \frac{4}{5}$$

In the previous example we had $SX_\phi(10, r_0, r_1) = 1.197 > X(\tau_0, \tau_1)$ and firing the transaction $\mathsf{T} = \mathsf{A} : \mathsf{swap}(10, \tau_0, \tau_1)$ led to a positive gain.

Taking the values of the current example we have that $SX_\phi(10, r_0, r_1) = 0.57 < X(\tau_0, \tau_1)$, hence we expect a negative gain. By using the formula from Lemma 3.1.2 we can easily see that:

$$\begin{aligned} G_{\mathsf{A}}(\Gamma, \mathsf{T}) &= x \cdot (SX_\phi(10, r_0, r_1)P(\tau_1) - P(\tau_0)) \\ &= 10 \cdot \left(\frac{\phi r_1}{r_0 + \phi x} \cdot 5 - 4\right) \\ &= 10 \cdot \left(\frac{0.997 \cdot 40}{60 + 0.997 \cdot 10} \cdot 5 - 4\right) \\ &= 10 \cdot (0.57 \cdot 5 - 4) \\ &= -11.50 \end{aligned}$$

3.2 General properties of the swap function

In the previous section, we defined properties regarding the gain of a user from a swap transaction. In this section we define some key properties regarding swap functions and what they imply on the AMMs.

Output boundedness guarantees that an AMM has always enough output tokens τ_1 to send to the user who performs a $\mathsf{swap}(x, \tau_0, \tau_1)$.

Definition 3.2.1 ▶ Output-boundedness

A swap rate function SX_ϕ is *output-bounded* when, for all x, r_0, r_1 such that $x \geq 0$ and $r_0, r_1 > 0$:

$$x \cdot SX_\phi(x, r_0, r_1) < r_1$$

In other terms, this ensures that for any SX_ϕ that satisfies this definition, the reserves of an already existing AMM will never be drained.

Monotonicity relates the parameters of the swap rate function with its output. In particular, the output decreases if we decrease the amount of swapped tokens x, if we decrease the reserves of τ_0 or if we increase the reserves of τ_1.

Definition 3.2.2 ▶ Strict monotonicity

A swap rate function SX_ϕ is *strictly monotonic* when, for $i \in \{0, 1, 2\}$ and $\lhd_i \in \{<, \leq\}$:

$$x' \lhd_0 x,\ r'_0 \lhd_1 r_0,\ r_1 \lhd_2 r'_1 \implies SX_\phi(x, r_0, r_1) \lhd_3 SX_\phi(x', r'_0, r'_1)$$

where:

$$\lhd_3 = \begin{cases} \leq & \text{if } \lhd_i = \leq \text{ for } i \in \{0, 1, 2\} \\ < & \text{otherwise} \end{cases}$$

This means that, for example, given a fixed input amount x and a fixed reserve of τ_0 r_0, a lower amount of reserves of τ_1 r_1 results in a higher swap rate output. This is crucial in proving some of the lemmas presented further in the paper.

The next crucial definition we want to introduce is the one of additivity, i.e. what is the output of two combined swaps. However, before doing so, we have to introduce a few auxiliary terms, since our definition of additivity slightly differs from the one introduced in [8]. The definition below, for example, denotes the factor from which our definition of additivity differs from the original one.

Definition 3.2.3 ▶ ζ_ϕ

We define the function

$$\zeta_\phi(x, y, r_0, r_1) = \frac{\Big((\phi r_1 x)(r_0 + \phi x + \phi y) + (\phi r_1 r_0 y)\Big) \cdot (r_0 + x + \phi y)}{(r_0 + \phi x + \phi y) \cdot \Big((\phi r_1 x)(r_0 + x + \phi y) + (\phi r_1 r_0 y)\Big)}$$

where $x, y, r_0, r_1 \in \mathbb{R} > 0$

Since this factor can be a bit too complex to work with, especially in some proofs, we

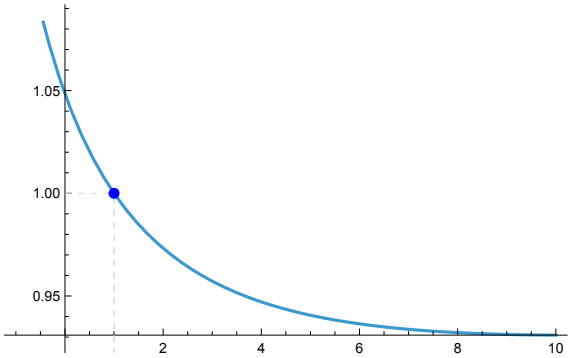

Figure 3.1: $\zeta_\phi(x, y, r_0, r_1)$ plot

introduce an equivalent factor in the lemma below, which is slightly easier to work with in some cases.

Lemma 3.2.4 ▶ ζ_ϕ reduced form

$\forall\, x, y, r_0, r_1 \in \mathbb{R} > 0,$

$$\zeta_\phi(x, y, r_0, r_1) = \frac{(x+y)(r_0+\phi x)(r_0+x+\phi y)}{(r_0+\phi x+\phi y)(x^2+r_0 x+\phi xy+r_0 y)}$$

Proof.

$$\begin{aligned}
\zeta_\phi(x, y, r_0, r_1) &= \frac{\Big((\phi r_1 x)(r_0+\phi x+\phi y)+(\phi r_1 r_0 y)\Big)\cdot(r_0+x+\phi y)}{(r_0+\phi x+\phi y)\cdot\Big((\phi r_1 x)(r_0+x+\phi y)+(\phi r_1 r_0 y)\Big)} \\
&= \frac{\Big(\phi r_0 r_1 x+\phi^2 r_1 x^2+\phi^2 r_1 xy+\phi r_0 r_1 y\Big)\cdot(r_0+x+\phi y)}{(r_0+\phi x+\phi y)\cdot\Big(\phi r_0 r_1 x+\phi r_1 x^2+\phi^2 r_1 xy+\phi r_0 r_1 y\Big)} \\
&= \frac{\Big(\phi r_1 x(r_0+\phi x)+\phi r_1 y(r_0+\phi x)\Big)\cdot(r_0+x+\phi y)}{(r_0+\phi x+\phi y)\cdot\Big(\phi r_1(r_0 x+x^2+\phi xy+r_0 y)\Big)} \\
&= \frac{\phi r_1\Big(x(r_0+\phi x)+y(r_0+\phi x)\Big)\cdot(r_0+x+\phi y)}{\phi r_1(r_0+\phi x+\phi y)(r_0 x+x^2+\phi xy+r_0 y)} \\
&= \frac{(x+y)(r_0+\phi x)(r_0+x+\phi y)}{(r_0+\phi x+\phi y)(x^2+r_0 x+\phi xy+r_0 y)}
\end{aligned}$$

□

It is also worth noting how this form is evidently less complex, as it does not depend at all on r_1. Next, it can be interesting to understand which values this term can assume, if it is always positive, negative or so on.

Example 3.2.5. The Figure Figure 3.1 shows the plot of $\zeta_\phi(x, y, r_0, r_1)$ in the *y-axis* and ϕ on the *x-axis*, with $x = 10$, $y = 3$, $r_0 = 40$ and $r_1 = 60$. While it is hard understand

from this plot that the following result does not depend on x, y, r_0 or r_1, we can see that in this case $\zeta_\phi(x, y, r_0, r_1)$ is strictly greater than 1 when $fee < 1$. This result is better formalized in the lemma below.

Lemma 3.2.6 ▶ $\zeta_\phi > 1$

$$\forall\, x, y, r_0, r_1 \in \mathbb{R} > 0,\ \phi < 1 \implies \zeta_\phi(x, y, r_0, r_1) > 1$$

Proof. Assume that $\phi < 1$, then:

$$\begin{aligned}
& \zeta_\phi(x, y, r_0, r_1) > 1 \\
\iff & \zeta_\phi(x, y, r_0, r_1) - 1 > 0 \\
\iff & \frac{(x+y)(r_0+\phi x)(r_0+x+\phi y)}{(r_0+\phi x+\phi y)(x^2+r_0x+\phi xy+r_0y)} - 1 > 0 && \text{By 3.2.4} \\
\iff & (x+y)(r_0+\phi x)(r_0+x+\phi y) \\
& - (r_0+\phi x+\phi y)(x^2+r_0x+\phi xy+r_0y) > 0 && \text{Denum} > 0 \\
\iff & r_0xy + x(r_0+\phi x)(r_0+x+\phi y) + r_0y(r_0+\phi y) + \phi xy(r_0+x+\phi y) \\
& - \phi r_0xy - \phi x(x^2+r_0x+\phi xy) - (r_0+\phi y)(x^2+r_0x+\phi xy+r_0y) \\
& > 0 \\
\iff & r_0xy(1-\phi) + (r_0+\phi x)(x^2+r_0x+\phi xy) + \phi y(x^2+r_0x+\phi xy) \\
& + r_0y(r_0+\phi y) - \phi x(x^2+r_0x+\phi xy) - r_0y(r_0+\phi y) \\
& - (r_0+\phi y)(x^2+r_0x+\phi xy) > 0 \\
\iff & r_0xy(1-\phi) + (r_0+\phi x)(x^2+r_0x+\phi xy) + \phi y(x^2+r_0x+\phi xy) \\
& - \phi x(x^2+r_0x+\phi xy) - (r_0+\phi y)(x^2+r_0x+\phi xy) > 0 \\
\iff & r_0xy(1-\phi) + (x^2+r_0x+\phi xy)(r_0+\phi x+\phi y-\phi x-r_0-\phi y) > 0 \\
\iff & r_0xy(1-\phi) > 0 \\
\iff & (1-\phi) > 0 \\
\iff & \phi < 1
\end{aligned}$$

□

It is also immediate to see that $\phi = 1$ *iff* $\zeta_\phi(x, y, r_0, r_1) = 1$. We omit the proof as it would have the exact same steps. This is of particular interest given our own definition of additivity, which slightly differs from the one presented in [8]:

Definition 3.2.7 ▶ Additivity

If a swap rate function SX_ϕ is *output-bounded* then it is also *additive* when:

$$SX_\phi(x + y, r_0, r_1) = \frac{\alpha x + \beta y}{x + y} \cdot \zeta_\phi(x, y, r_0, r_1)$$

where

$$\alpha = SX_\phi(x, r_0, r_1),\ \beta = SX_\phi(y, r_0 + x, r_1 - \alpha x)$$

Note how the term $\zeta_\phi(x, y, r_0, r_1)$ multiplies the fraction, and we know that $\zeta_\phi(x, y, r_0, r_1) = 1$ *iff* $\phi = 1$. Hence, we find that this definition of additivity falls back to the one defined in

[8] in the specific case where $\phi = 1$. This is also something expected, as the same happens with the constant product swap rate (Def. 2.4.1). Since we know from Lemma 3.2.6 that this factor is always greater than one when $\phi < 1$, this means that in a model that uses fees and with a swap rate function that satisfies this definition, a user is more likely to swap larger sums of tokens rather than splitting this amount into two. We give an empirical example of this claim later when we prove properties about the constant product swap rate.

The following theorem shows how additivity and gain are related to each other. In particular, the gain of two consecutive swap actions is the sum of their individual gains plus a term ε_ϕ.

Theorem 3.2.8 ▶ Additivity of swap gain

Let $\Gamma = \mathsf{A}[\sigma_\mathsf{A}] \mid \{r_0 : \tau_0, r_1 : \tau_1\} \mid \Delta$ be such that $S_\Gamma\{\tau_0, \tau_1\} > 0$ and $\sigma_\mathsf{A}\{\tau_0, \tau_1\} < S_\Gamma\{\tau_0, \tau_1\}$. Let $\mathsf{T}(x) = \mathsf{A} : \mathsf{swap}(x, \tau_0, \tau_1)$ and let Γ' and Γ'' be such that, $\Gamma \xrightarrow{\mathsf{T}(x_0)} \Gamma'$ and $\Gamma' \xrightarrow{\mathsf{T}(x_1)} \Gamma''$.
If SX_ϕ is *output-bounded* and *additive*, and $\phi < 1$ then:

$$G_\mathsf{A}(\Gamma, \mathsf{T}(x_0 + x_1)) \;=\; G_\mathsf{A}(\Gamma, \mathsf{T}(x_0)) + G_\mathsf{A}(\Gamma', \mathsf{T}(x_1)) + \varepsilon_\phi$$

where:

$$\begin{aligned}
\varepsilon_\phi &= P(\tau_1)(\zeta_\phi(x_0, x_1, r_0, r_1) - 1)(\alpha x_0 + \beta x_1)\eta \\
\alpha &= SX_\phi(x_0, r_0, r_1) \\
\beta &= SX_\phi(x_1, r_0 + x_0, r_1 - \alpha x_0) \\
\eta &= 1 - \frac{\sigma_\mathsf{A}\{\tau_0, \tau_1\}}{S_\Gamma\{\tau_0, \tau_1\}}
\end{aligned}$$

Proof. By Definition 3.2.7 we have that

$$\gamma = SX_\phi(x_0 + x_1, r_0, r_1) = \frac{\alpha x_0 + \alpha x_1}{x_0 + x_1} \cdot \zeta_\phi(x_0, x_1, r_0, r_1)$$

For abbreviation, let

$$\begin{aligned}
\mathcal{Z} &= \zeta_\phi(x_0, x_1, r_0, r_1) \\
\eta &= \frac{\sigma_\mathsf{A}\{\tau_0, \tau_1\}}{S_\Gamma\{\tau_0, \tau_1\}}
\end{aligned}$$

Hence,

$$\begin{aligned}
& G_\mathsf{A}(\Gamma, \mathsf{T}(x_0 + x_1)) - G_\mathsf{A}(\Gamma, \mathsf{T}(x_0)) \\
&= \eta\left(\gamma(x_0 + x_1)P(\tau_1) - (x_0 + x_1)P(\tau_0) - \alpha x_0 P(\tau_1) + x_0 P(\tau_0)\right) \\
&= \eta\left((\gamma(x_0 + x_1) - \alpha x_0)P(\tau_1) - x_1 P(\tau_0)\right) \\
&= \eta\left((\mathcal{Z}\alpha x_0 + \mathcal{Z}\beta x_1 - \alpha x_0)P(\tau_1) - x_1 P(\tau_0)\right) \\
&= \eta\left(\mathcal{Z}\alpha x_0 P(\tau_1) + \mathcal{Z}\beta x_1 P(\tau_1) - \alpha x_0 P(\tau_1) - x_1 P(\tau_0)\right) \\
&= \eta(\mathcal{Z}\alpha x_0 P(\tau_1) + \mathcal{Z}\beta x_1 P(\tau_1) - \alpha x_0 P(\tau_1) - x_1 P(\tau_0) \\
&\quad + \beta x_1 P(\tau_1) - \beta x_1 P(\tau_1))
\end{aligned}$$

$$
\begin{aligned}
&= \eta((\beta x_1 P(\tau_1) - x_1 P(\tau_0)) + (\mathcal{Z}\alpha x_0 P(\tau_1) + \mathcal{Z}\beta x_1 P(\tau_1) \\
&\quad - \alpha x_0 P(\tau_1) - \beta x_1 P(\tau_0))) \\
&= G_{\mathsf{A}}(\Gamma', \mathsf{T}(x_1)) + P(\tau_1)(\mathcal{Z} - 1)(\alpha x_0 + \beta x_1)\eta && \text{(by Def. 3.1.1)} \\
&= G_{\mathsf{A}}(\Gamma', \mathsf{T}(x_1)) + \varepsilon_\phi && \text{(by Def. of } \varepsilon_\phi)
\end{aligned}
$$

□

Notice how in ε_ϕ, the term $(\zeta_\phi(x_0, x_1, r_0, r_1) - 1)$ multiplies all the other terms. As stated before, we know that $\zeta_\phi(x_0, x_1, r_0, r_1) = 1 \iff \phi = 1$. Hence, if $\phi = 1$ this result falls back to the one presented in [8], where $G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0 + x_1)) = G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + G_{\mathsf{A}}(\Gamma', \mathsf{T}(x_1))$, since $\varepsilon_\phi = 0$.

From the definition of ε_ϕ it is also worth noting that this term is always positive under the lemma's assumptions, since all the terms are strictly positive and $\zeta_\phi(x_0, x_1, r_0, r_1) > 1$ from Lemma 3.2.6.

3.3 Properties of the costant-product swap function

In this section we study in depth the constant product swap rate (Def 2.4.1). As previously stated, this is the swap rate function that is mostly used in real-world AMMs implementations (such as Uniswap v2 [13]), hence it is of particular interest analyzing the properties satisfied by this function.

First of all, it is output-bounded, so AMMs that implement this function are guaranteed to have non-negative reserves upon any valid swap transaction.

Lemma 3.3.1 ▶ Constant product *output-boundedness*

The constant product swap rate function (Def. 2.4.1) is output bounded (Def. 3.2.1)

Proof.

$$
\begin{aligned}
& x \cdot SX_\phi(x, r_0, r_1) < r_1 \\
\iff & x \cdot \frac{\phi r_1}{r_0 + \phi x} < r_1 \\
\iff & x\phi r_1 < r_1 + (r_0 + \phi x) \\
\iff & \phi x r_1 < r_0 r_1 + \phi x r_1 \\
\iff & r_0 r_1 > 0 \\
\iff & \mathit{True}
\end{aligned}
$$

□

Then, the constant product swap rate function is also strictly monotonic. This also ensures that the internal rate is strictly monotonic, which is another key result later needed for the arbitrage proofs.

Lemma 3.3.2 ▶ Constant product *strict-monotonicity*

The constant product swap rate function (Def. 2.4.1) is strictly monotonic (Def. 3.2.2)

Proof. There are two possible cases either $x' < x \lor r'_0 < r_0 \lor r_1 < r'_1$ or not. in both cases, it is always true that $x' \leq x \land r'_0 \leq r_0 \land r_1 \leq r'_1$

1. $\neg(x' < x \lor r'_0 < r_0 \lor r_1 < r'_1)$:
   In this case we want to prove:

$$\begin{aligned} & SX_\phi(x, r_0, r_1) \leq SX_\phi(x', r'_0, r'_1) \\ \iff & \frac{\phi r_1}{r_0 + \phi x} \leq \frac{\phi r'_1}{r'_0 + \phi x'} \\ \iff & \phi r_1(r'_0 + \phi x') \leq \phi r'_1(r_0 + \phi x) \end{aligned}$$

   This is true iff

$$\phi r_1 \leq \phi r'_1 \iff r_1 \leq r'_1 \quad \text{By assm.}$$

   and

$$\begin{aligned} & r'_0 + \phi x' \leq r_0 + \phi x \\ \iff & \Big((r'_0 \leq r_0) \land (\phi x' \leq \phi x)\Big) && \text{By assm.} \\ \iff & x' \leq x && \text{By assm.} \end{aligned}$$

2. $x' < x \lor r'_0 < r_0 \lor r_1 < r'_1$:
   In this case we have three different cases:

   (a) Case $x' < x$:

$$\begin{aligned} & SX_\phi(x, r_0, r_1) < SX_\phi(x', r'_0, r'_1) \\ \iff & \frac{\phi r_1}{r_0 + \phi x} < \frac{\phi r'_1}{r'_0 + \phi x'} \\ \iff & \phi r_1(r'_0 + \phi x') < \phi r'_1(r_0 + \phi x) \end{aligned}$$

   Since we know that $\phi r_1 \leq \phi r'_1$ and $r'_0 + \phi x' < r_0 + \phi x$ by assumptions, then the proof is finished.

   (b) Case $r'_0 < r_0$:

$$\begin{aligned} & SX_\phi(x, r_0, r_1) < SX_\phi(x', r'_0, r'_1) \\ \iff & \frac{\phi r_1}{r_0 + \phi x} < \frac{\phi r'_1}{r'_0 + \phi x'} \\ \iff & \phi r_1(r'_0 + \phi x') < \phi r'_1(r_0 + \phi x) \end{aligned}$$

   Since we know that $\phi r_1 \leq \phi r'_1$ and $r'_0 + \phi x' < r_0 + \phi x$ by assumptions, then the proof is finished.

(c) Case $r_1 < r'_1$:

$$\begin{aligned} & SX_\phi(x, r_0, r_1) < SX_\phi(x', r'_0, r'_1) \\ \iff & \frac{\phi r_1}{r_0 + \phi x} < \frac{\phi r'_1}{r'_0 + \phi x'} \\ \iff & \phi r_1(r'_0 + \phi x') < \phi r'_1(r_0 + \phi x) \end{aligned}$$

Since we know that $\phi r_1 < \phi r'_1$ and $r'_0 + \phi x' \leq r_0 + \phi x$ by assumptions, then the proof is finished.

□

Finally, we can also prove that the function is additive, based on the definition given above.

Lemma 3.3.3 ▶ Constant product *additivity*

The constant product swap rate function (Def. 2.4.1) is additive (Def. 3.2.7)

Proof. Let

$$\alpha = SX_\phi(x, r_0, r_1) = \frac{\phi r_1}{r_0 + \phi x}$$

$$\beta = SX_\phi(y, r_0 + x, r_1 - \alpha x) = \frac{\phi(r_1 - \alpha x)}{r_0 + x + \phi y} = \frac{\phi \cdot \left(r_1 - \frac{\phi r_1 x}{r_0 + \phi x}\right)}{r_0 + x + \phi y} = \frac{\phi\left(\frac{r_1(r_0 + \phi x) - \phi r_1 x}{r_0 + \phi x}\right)}{r_0 + x + \phi y}$$
$$= \frac{\phi(r_1 r_0 + \phi r_1 x - \phi r_1 x)}{(r_0 + \phi x)(r_0 + x + \phi y)} = \frac{\phi r_1 r_0}{(r_0 + \phi x)(r_0 + x + \phi y)}$$

Then,

$$\begin{aligned} \frac{\alpha x + \beta y}{x + y} \cdot z &= \frac{1}{x + y}\left(\frac{\phi r_1 x}{r_0 + \phi x} + \frac{\phi r_1 r_0 y}{(r_0 + \phi x)(r_0 + x + \phi y)}\right) \cdot z \\ &= \frac{1}{x + y}\left(\frac{\phi r_0 r_1 x + \phi r_1 x^2 + \phi^2 r_1 x y + \phi r_0 r_1 y}{(r_0 + \phi x)(r_0 + x + \phi y)}\right) \cdot z \\ &= \frac{1}{x + y}\left(\frac{(\phi r_1 x)(r_0 + x + \phi y) + (\phi r_1 r_0 y)}{(r_0 + \phi x)(r_0 + x + \phi y)}\right) \cdot \\ &\quad \frac{\Big((\phi r_1 x)(r_0 + \phi x + \phi y) + (\phi r_1 r_0 y)\Big) \cdot (r_0 + x + \phi y)}{(r_0 + \phi x + \phi y) \cdot \Big((\phi r_1 x)(r_0 + x + \phi y) + (\phi r_1 r_0 y)\Big)} \\ &= \frac{1}{x + y} \frac{(\phi r_1 x)(r_0 + \phi x + \phi y) + (\phi r_1 r_0 y)}{(r_0 + \phi x)(r_0 + \phi x + \phi y)} \\ &= \frac{1}{x + y} \frac{\phi r_0 r_1 x + \phi^2 r_1 x^2 + \phi^2 r_1 x y + \phi r_1 r_0 y}{(r_0 + \phi x)(r_0 + \phi x + \phi y)} \\ &= \frac{1}{x + y} \frac{\phi r_1(r_0 x + \phi x^2 + \phi x y + r_0 y)}{(r_0 + \phi x)(r_0 + \phi x + \phi y)} \\ &= \frac{1}{x + y} \frac{\phi r_1(r_0 + \phi x)(x + y)}{(r_0 + \phi x)(r_0 + \phi x + \phi y)} \end{aligned}$$

$$= \frac{\phi r_1}{r_0 + \phi(x + y)}$$
$$= SX_\phi(x + y, r_0, r_1)$$

□

At first sight, the reader might think that the gain given by x tokens on an AMM that uses the constant product swap rate function is the same as the sum of the gains of two consecutive swaps, where the sum of the swapped tokens is equal to x. However, we just proved that the constant product satisfies the definition of additivity given in Def. 3.2.7. So, if the trading fee ϕ is less than one, and the user performing the swap has no minted token, then it also means that the function satisfies all the assumptions of Theorem 3.2.8.

Example 3.3.4. Let $\Gamma = \mathsf{A}[300 : \tau_0, 200 : \tau_1] \mid \{400 : \tau_0, 600 : \tau_1\}$. Now suppose that the prices of τ_0 and τ_1 are $P(\tau_0) = 4$ and $P(\tau_1) = 5$. Let $\mathsf{T}_0 = \mathsf{A} : \mathsf{swap}(100, \tau_0, \tau_1)$ and suppose that SX_ϕ is the constant product swap rate and that $\phi = 0.997$. If A were to fire the transaction T_0, i.e. $\Gamma \xrightarrow{\mathsf{T}_0} \Gamma'$, A would get a total of $y = 100 * SX_\phi(100, 400, 600) = 119.712\ \tau_1$ tokens, and the resulting state would be

$$\Gamma' = \mathsf{A}[200 : \tau_0, 319.712 : \tau_1] \mid \{500 : \tau_0, 480.288 : \tau_1\}$$

Now, suppose that A instead of firing T_0 directly decides to split this transaction into two transactions, $\mathsf{T}_1 = \mathsf{A} : \mathsf{swap}(40, \tau_0, \tau_1)$ and $\mathsf{T}_2 = \mathsf{A} : \mathsf{swap}(60, \tau_0, \tau_1)$. The user A decides to fire T_1 first, so $\Gamma \xrightarrow{\mathsf{T}_1} \Gamma_1 \xrightarrow{\mathsf{T}_2} \Gamma_2$ where

$$\Gamma_1 = \mathsf{A}[260 : \tau_0, 254.397 : \tau_1] \mid \{460 : \tau_0, 545.603 : \tau_1\}$$

and

$$\Gamma_2 = \mathsf{A}[200 : \tau_0, 319.696 : \tau_1] \mid \{500 : \tau_0, 480.304 : \tau_1\}$$

Clearly, we can see that the split did not benefit A, as he received a tiny bit less tokens that what he would have received by firing the transaction T_0 directly instead of splitting it in T_1 and T_2. Consequently, the gain of A will also be less by splitting T_0 in two transactions, backing up what we already stated in Theorem 3.2.8

3.4 Internal Rate Properties

In this section we present some lemmas that describe the relationships between the internal exchange rate $X_\Gamma(\tau_0, \tau_1)$, the exchange rate $X(\tau_0, \tau_1)$ and the swap rate $SX_\phi(x, r_0, r_1)$ particularly for the constant product swap rate. This is of keen interest since it allows us to reason over the gain of the user, combining these results with Lemma 3.1.4 and helping us reason on the conditions that incentivize a user to perform a swap.

First, we analyze the relationship between the swap rate and the internal rate.

Lemma 3.4.1 ▶ Swap rate vs Internal rate

Let $\Gamma = \mathsf{A}[\sigma_{\mathsf{A}}] \mid \{r_0 : \tau_0, r_1 : \tau_1\} \mid \Delta$ and $\Gamma \xrightarrow{\mathsf{A}:\mathsf{swap}(x,\tau_0,\tau_1)} \Gamma'$. Then, if SX_ϕ is the constant product swap rate and $\phi \leq 1$:

$$X_{\Gamma'}(\tau_0, \tau_1) < SX_\phi(x, r_0, r_1) < X_\Gamma(\tau_0, \tau_1) \tag{3.2}$$

Proof. We split the proof in two parts:

- $SX_\phi(x_0, r_0, r_1) < X_\Gamma(\tau_0, \tau_1)$:

$$\begin{aligned} X_\Gamma(\tau_0, \tau_1) &= \lim_{z \to 0} SX_\phi(z, r_0, r_1) && \text{(by Def. of X)} \\ &> SX_\phi(x_0, r_0, r_1) && \text{Strict. Mono.} \end{aligned}$$

- $X_{\Gamma'}(\tau_0, \tau_1) < SX_\phi(x_0, r_0, r_1)$:

  Let $\alpha = SX_\phi(x_0, r_0, r_1)$.

  Before continuing the proof, lets first prove that

$$\frac{\phi(r_1 - \alpha x_0)}{r_0 + x_0} < \frac{\phi r_1}{r_0 + \phi x_0} \tag{3.3}$$

  Proof.

$$\begin{aligned} &\frac{\phi(r_1 - \alpha x_0)}{r_0 + x_0} < \frac{\phi r_1}{r_0 + \phi x_0} \\ \iff &\phi(r_1 - \alpha x_0)(r_0 + \phi x_0) < \phi r_1(r_0 + x_0) \\ \iff &\phi r_1(r_0 + \phi x_0) - \phi\alpha x_0(r_0 + \phi x_0) - \phi r_1(r_0 + \phi x_0) < 0 \\ \iff &\phi r_1(\phi x_0 - x_0) - \phi\alpha x_0(r_0 + \phi x_0) < 0 \\ \iff &\phi r_1 x_0(\phi - 1) - \phi\alpha x_0(r_0 + \phi x_0) < 0 \end{aligned}$$

  Since we know that $\phi\alpha x_0(r_0 + \phi x_0) > 0$ because all the terms are strictly positive, and $\phi r_1 x_0(\phi - 1) \leq 0 \iff \phi \leq 1$, then we finish the proof. □

  Now we can continue the proof:

$$\begin{aligned} X_{\Gamma'}(\tau_0, \tau_1) &= \lim_{z \to 0} SX_\phi(z, r_0 + x_0, r_1 - \alpha x_0) && \text{(by Def. of X)} \\ &= \frac{\phi(r_1 - \alpha x_0)}{r_0 + x_0} \\ &< \frac{\phi r_1}{r_0 + x_0} && \text{by Eq 3.3} \\ &= SX_\phi(x_0, r_0, r_1) \end{aligned}$$

□

So we can see how the swap rate sits right between the internal rates computed in the AMM state before and after performing the swap transaction. It is also worth noting how this lemma is valid also when $\phi = 1$, which means that it would be valid also in the model presented in [8].

Finally, we also want to analyze how the internal and external swap rate relate to each other. In particular, we consider a valid swap transaction and suppose that after this transaction, the external swap rate and the internal one are equal. This is a key assumption that we use thoroughly when reasoning about the arbitrage later on.

LEAN

Lemma 3.4.2 ▶ Split Internal rate vs External rate

Let $\Gamma = \mathsf{A}[\sigma_\mathsf{A}] \mid \{r_0 : \tau_0, r_1 : \tau_1\} \mid \Delta$ and $\mathsf{T}(x) = \mathsf{A} : \mathsf{swap}(x, \tau_0, \tau_1)$.
Let $x_0 \in \mathbb{R}_{>0}$ be such that:

$$X_{\Gamma'}(\tau_0, \tau_1) = X(\tau_0, \tau_1) \qquad \text{where } \Gamma \xrightarrow{\mathsf{T}(x_0)} \Gamma' \tag{3.4}$$

Let $x_1,\ x_2 \in \mathbb{R}_{>0}$ be such that $x_0 = x_1 + x_2$. Then, if SX_ϕ is the constant product swap rate (2.4.1) and $\phi < 1$:

$$X(\tau_0, \tau_1) < X_{\Gamma_2}(\tau_0, \tau_1) \qquad \text{where } \Gamma \xrightarrow{\mathsf{T}(x_1)} \Gamma_1 \xrightarrow{\mathsf{T}(x_2)} \Gamma_2 \tag{3.5}$$

Proof. Consider

$$\begin{aligned}
y_1 &= SX_\phi(x_1, r_0, r_1) \cdot x_1 \\
&= \frac{\phi r_1 x_1}{r_0 + \phi x_1} \\
y_2 &= SX_\phi(x_2, r_0 + x_1, r_1 - y_1) \cdot x_2 \\
&= \frac{\phi r_0 r_1 x_2}{(r_0 + \phi x_1)(r_0 + x_1 + \phi x_2)} \\
y_3 &= SX_\phi(x1 + x_2, r_0 + x_1 + x_2, r_1 - (y_1 + y_2)) \cdot (x_1 + x_2) \\
&= SX_\phi(x_0, r_0 + x_0, r_1 - (y_1 + y_2)) \cdot x_0 \\
&= \frac{\phi r_1 (x_1 + x_2)}{r_0 + \phi x_1 + \phi x_2}
\end{aligned}$$

Then,

$$\begin{aligned}
X_{\Gamma_2}(\tau_0, \tau_1) &= \lim_{z \to 0} SX_\phi(z, r_0 + x_0, r_1 - (y_1 + y_2)) && \text{(By Def.)} \\
&> \lim_{z \to 0} SX_\phi(z, r_0 + x_0, r_1 - y_3) && \text{(By strict mono. proof below)} \\
&= X_{\Gamma'}(\tau_0, \tau_1) = X(\tau_0, \tau_1) && \text{(By. Def. and Hyp.)}
\end{aligned}$$

We know that $z \leq z$ and $r_0 + x_0 \leq r_0 + x_0$, hence we only have to prove that $r_1 - (y_1 + y_2) > r_1 - y_3$:

$$\begin{aligned}
& r_1 - (y_1 + y_2) > r_1 - y_3 \\
\iff & y_1 + y_2 - y_3 < 0
\end{aligned}$$

$$
\begin{aligned}
&\iff \frac{\phi r_1 x_1}{r_0 + \phi x_1} + \frac{\phi r_0 r_1 x_2}{(r_0 + \phi x_1)(r_0 + x_1 + \phi x_2)} - \frac{\phi r_1 (x_1 + x_2)}{r_0 + \phi x_1 + \phi x_2} < 0 \\
&\iff \frac{\phi r_1 x_1 (r_0 + x_1 + \phi x_2)(r_0 + \phi x_1 + \phi x_2) + \phi r_0 r_1 x_2 (r_0 + \phi x_1 + \phi x_2)}{(r_0 + \phi x_1)(r_0 + x_1 + \phi x_2)(r_0 + \phi x_1 + \phi x_2)} \\
&\quad - \frac{\phi r_1 x_1 (r_0 + \phi x_1)(r_0 + x_1 + \phi x_2) + \phi r_1 x_2 (r_0 + \phi x_1)(r_0 + x_1 + \phi x_2)}{(r_0 + \phi x_1)(r_0 + x_1 + \phi x_2)(r_0 + \phi x_1 + \phi x_2)} < 0 \\
&\iff \phi r_1 x_1 (r_0 + x_1 + \phi x_2)(r_0 + \phi x_1 + \phi x_2) + \phi r_0 r_1 x_2 (r_0 + \phi x_1 + \phi x_2) \\
&\quad - \phi r_1 x_1 (r_0 + \phi x_1)(r_0 + x_1 + \phi x_2) - \phi r_1 x_2 (r_0 + \phi x_1)(r_0 + x_1 + \phi x_2) < 0 \\
&\iff \phi r_1 x_1 (r_0 + x_1 + \phi x_2)((r_0 + \phi x_1 + \phi x_2) - (r_0 + \phi x_1)) + \phi r_0 r_1 x_2 (r_0 + \phi x_2) \\
&\quad + \phi^2 r_0 r_1 x_1 x_2 - \phi r_0 r_1 x_1 x_2 - \phi r_0 r_1 x_2 (r_0 + \phi x_2) - \phi^2 r_1 x_1 x_2 (r_0 + x_1 + \phi x_2) < 0 \\
&\iff \phi^2 r_1 x_1 x_2 (r_0 + x_1 + \phi x_2) + \phi r_0 r_1 x_2 (r_1 + \phi x_2) + \phi^2 r_0 r_1 x_1 x_2 - \phi r_0 r_1 x_1 x_2 \\
&\quad - \phi r_0 r_1 x_2 (r_0 + \phi x_2) - \phi^2 r_1 x_1 x_2 (r_0 + x_1 + \phi x_2) < 0 \\
&\iff \phi^2 r_0 r_1 x_1 x_2 - \phi r_0 r_1 x_1 x_2 < 0 \\
&\iff \phi r_0 r_1 x_1 x_2 (\phi - 1) < 0 \\
&\iff \phi - 1 < 0 \\
&\iff \phi < 1
\end{aligned}
$$

□

So in other terms, if we split the x_0 that brings the internal swap rate equal to the external one, and perform the two swaps, then the internal rate will be strictly greater than the external one. This is a result that we expected, since we already proved in Lemma 3.3.3 that splitting a swap transaction in two different swaps brings the AMM into a different state as well.

3.5 Arbitrage

In this section we want to present the final results of this paper. In particular, our main goal is to find the optimal amount of tokens that a user can swap given a certain state of the AMM that maximizes his gain. A first attempt at doing this could be mimicking the result found in [8] , i.e. the optimal value is the one that brings the internal rate equal to the external one. We often refer to this as the AMM reaching the "Equilibrium".

The first challenge that arises is finding this particular value, which we specify in the Theorem below.

LEAN

Theorem 3.5.1 ▶ Equilibrium value

Let $\Gamma = \mathsf{A}[\sigma_{\mathsf{A}}] \mid \{r_0 : \tau_0, r_1 : \tau_1\}$, and let:

$$x_0 = \frac{-\sqrt{P(\tau_0)}r_0(1+\phi) + \sqrt{r_0}\sqrt{P(\tau_0)r_0(-1+\phi)^2 + 4P(\tau_1)r_1\phi^2}}{2\sqrt{P(\tau_0)}\phi} \tag{3.6}$$

If SX_ϕ is the constant product swap rate, $\mathsf{T}(x_0) = \mathsf{A} : \mathsf{swap}(x_0, \tau_0, \tau_1)$ is enabled in Γ and $\phi < 1$ then:

$$X(\tau_0, \tau_1) = X_{\Gamma'}(\tau_0, \tau_1) \qquad \text{where } \Gamma \xrightarrow{\mathsf{T}(x_0)} \Gamma'$$

Proof. Before starting the proof, we rewrite some terms:

$$\begin{aligned}
x_0 + r_0 &= \frac{-\sqrt{P(\tau_0)}r_0(1+\phi) + \sqrt{r_0}\sqrt{P(\tau_0)r_0(-1+\phi)^2 + 4P(\tau_1)r_1\phi^2}}{2\sqrt{P(\tau_0)}\phi} + r_0 \\
&= \frac{-\sqrt{P(\tau_0)}r_0(1+\phi) + \sqrt{r_0}\sqrt{P(\tau_0)r_0(-1+\phi)^2 + 4P(\tau_1)r_1\phi^2} + 2\sqrt{P(\tau_0)}\phi r_0}{2\sqrt{P(\tau_0)}\phi} \\
&= \frac{-\sqrt{P(\tau_0)}r_0(1-\phi) + \sqrt{r_0}\sqrt{P(\tau_0)r_0(-1+\phi)^2 + 4P(\tau_1)r_1\phi^2}}{2\sqrt{P(\tau_0)}\phi} \\
&= \frac{\sqrt{P(\tau_0)}r_0(\phi-1) + \sqrt{r_0}\sqrt{P(\tau_0)r_0(-1+\phi)^2 + 4P(\tau_1)r_1\phi^2}}{2\sqrt{P(\tau_0)}\phi}
\end{aligned} \tag{3.7}$$

$$\begin{aligned}
\phi x_0 + r_0 &= \phi \cdot \frac{-\sqrt{P(\tau_0)}r_0(1+\phi) + \sqrt{r_0}\sqrt{P(\tau_0)r_0(-1+\phi)^2 + 4P(\tau_1)r_1\phi^2}}{2\sqrt{P(\tau_0)}\phi} + r_0 \\
&= \frac{-\sqrt{P(\tau_0)}r_0(1+\phi) + \sqrt{r_0}\sqrt{P(\tau_0)r_0(-1+\phi)^2 + 4P(\tau_1)r_1\phi^2} + 2\sqrt{P(\tau_0)}r_0}{2\sqrt{P(\tau_0)}} \\
&= \frac{-\sqrt{P(\tau_0)}r_0(\phi-1) + \sqrt{r_0}\sqrt{P(\tau_0)r_0(-1+\phi)^2 + 4P(\tau_1)r_1\phi^2}}{2\sqrt{P(\tau_0)}}
\end{aligned} \tag{3.8}$$

And let

$$m = \sqrt{r_0}\sqrt{P(\tau_0)r_0(-1+\phi)^2 + 4P(\tau_1)r_1\phi^2}$$
$$\alpha = SX_\phi(x_0, r_0, r_1) = \frac{\phi r_1}{r_0 + \phi x_0}$$

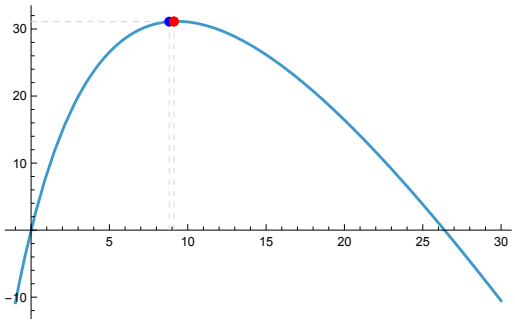

Figure 3.2: $G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0))$

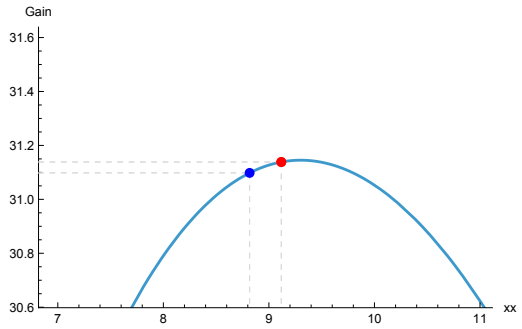

Figure 3.3: $G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0))$ zoomed

Then,

$$
\begin{aligned}
X_{\Gamma'}(\tau_0, \tau_1) &= \lim_{z \to 0} SX_{\phi}(z, r_0 + x_0, r_1 - \alpha x_0) && \text{(By Def.)} \\
&= \frac{\phi(r_1 - x_0 \cdot \frac{\phi r_1}{r_0 + \phi x_0})}{r_0 + x_0} \\
&= \frac{\phi(\frac{r_0 r_1 + \phi r_1 x_0 - \phi r_1 x_0}{r_0 + \phi x_0})}{r_0 + x_0} \\
&= \frac{\phi r_0 r_1}{(r_0 + x_0)(r_0 + \phi x_0)} \\
&= \frac{\phi r_0 r_1}{(\frac{\sqrt{P(\tau_0)} r_0(\phi - 1) + m}{2\sqrt{P(\tau_0)}\phi})(\frac{-\sqrt{P(\tau_0)} r_0(\phi - 1) + m}{2\sqrt{P(\tau_0)}})} && \text{(By Subst.)} \\
&= \frac{\phi r_0 r_1}{\frac{(\sqrt{P(\tau_0)} r_0(\phi - 1) + m)(-\sqrt{P(\tau_0)} r_0(\phi - 1) + m)}{4P(\tau_0)\phi}} \\
&= \frac{\phi r_0 r_1 (4P(\tau_0)\phi)}{m^2 - (\sqrt{P(\tau_0)} r_0(\phi - 1))^2} \\
&= \frac{\phi r_0 r_1 (4P(\tau_0)\phi)}{r_0(P(\tau_0) r_0(\phi - 1)^2 + 4P(\tau_1) r_1 \phi^2) - P(\tau_0) r_0^2 (\phi - 1)^2} \\
&= \frac{\phi r_0 r_1 (4P(\tau_0)\phi)}{P(\tau_0) r_0^2 (\phi - 1)^2 + 4P(\tau_1) r_0 r_1 \phi^2 - P(\tau_0) r_0^2 (\phi - 1)^2} \\
&= \frac{\phi r_0 r_1 (4P(\tau_0)\phi)}{4P(\tau_1) r_0 r_1 \phi^2} \\
&= \frac{P(\tau_0)}{P(\tau_1)}
\end{aligned}
$$

□

Hence we know that this value, given the appropriate conditions, brings the AMM to the equilibrium. However, we are not sure, or better, we have not proved yet, that this is the only value that satisfies this condition. The following lemma states exactly this.

Lemma 3.5.2 ▶ Balance swap value uniqueness

Let $\Gamma = \mathsf{A}[\sigma_{\mathsf{A}}] \mid \{r_0 : \tau_0, r_1 : \tau_1\}$ and $\mathsf{T}(x_0) = \mathsf{A} : \mathsf{swap}(x_0, \tau_0, \tau_1)$ be enabled in Γ. If SX_ϕ is the constant product swap rate and

$$X(\tau_0, \tau_1) = X_{\Gamma'}(\tau_0, \tau_1) \qquad \text{where } \Gamma \xrightarrow{\mathsf{A}:\mathsf{swap}(x_0,\tau_0,\tau_1)} \Gamma' \tag{3.9}$$

then

$$\neg\exists\, x_1 \in \mathbb{R}_{>0}, x_1 \neq x_0 \ \wedge\ X(\tau_0, \tau_1) = X_{\Gamma''}(\tau_0, \tau_1) \qquad \text{where } \Gamma \xrightarrow{\mathsf{A}:\mathsf{swap}(x_1,\tau_0,\tau_1)} \Gamma''$$

i.e. x_0 is unique.

Proof. By contradiction, assume that such x_1 exists, hence we know that:

$$x_1 \neq x_0 \ \wedge\ X(\tau_0, \tau_1) = X_{\Gamma''}(\tau_0, \tau_1) \tag{3.10}$$

Then, combining this with 3.9 by transitivity we have that

$$X_{\Gamma'}(\tau_0, \tau_1) = X_{\Gamma''}(\tau_0, \tau_1) \tag{3.11}$$

Let

$$\alpha = SX_\phi(x_0, r_0, r_1)$$
$$\beta = SX_\phi(x_1, r_0, r_1)$$

Further developing the above equality, we find that:

$$\begin{aligned}
& X_{\Gamma'}(\tau_0, \tau_1) = X_{\Gamma''}(\tau_0, \tau_1) \\
\iff & \lim_{z\to 0} SX_\phi(z, r_0 + x_0, r_1 - \alpha x_0) = \lim_{z\to 0} SX_\phi(z, r_0 + x_1, r_1 - \alpha x_1) \\
\iff & \frac{\phi\,(r_1 - \alpha x_0)}{r_0 + x_0} = \frac{\phi\,(r_1 - \beta x_1)}{r_0 + x_1} \\
\iff & \frac{\phi\,(r_0 r_1)}{(r_0 + \phi x_0)(r_0 + x_0)} = \frac{\phi\,(r_0 r_1)}{(r_0 + \phi x_1)(r_0 + x_1)} \\
\iff & (r_0 + \phi x_0)(r_0 + x_0) = (r_0 + \phi x_1)(r_0 + x_1) \\
\iff & r_0^2 + r_0 x_0 + r_0 \phi x_0 + \phi x_0^2 = r_0^2 + r_0 x_1 + r_0 \phi x_1 + \phi x_1^2 \\
\iff & r_0 x_0 + r_0 \phi x_0 + \phi x_0^2 = r_0 x_1 + r_0 \phi x_1 + \phi x_1^2 \\
\iff & r_0 x_0 (\phi + 1) + \phi x_0^2 = r_0 x_1 (\phi + 1) + \phi x_1^2 \\
\iff & r_0 x_0 (\phi + 1) + \phi x_0^2 - \left(r_0 x_1 (\phi + 1) + \phi x_1^2\right) = 0 \\
\iff & r_0 x_0 (\phi + 1) + \phi x_0^2 - r_0 x_1 (\phi + 1) - \phi x_1^2 = 0 \\
\iff & (r_0 x_0 - r_0 x_1)(\phi + 1) + \phi(x_0^2 - x_1^2) = 0 \\
\iff & r_0 (x_0 - x_1)(\phi + 1) + \phi(x_0^2 - x_1^2) = 0 \\
\iff & r_0 (x_0 - x_1)(\phi + 1) + \phi(x_0 - x_1)(x_0 + x_1) = 0
\end{aligned}$$

$$
\begin{aligned}
&\iff (x_0 - x_1)\,(r_0(\phi+1) + \phi(x_0 + x_1)) = 0 \\
&\iff x_0 - x_1 = 0 \vee r_0(\phi+1) + \phi(x_0 + x_1) = 0 \\
&\iff False \vee r_0(\phi+1) + \phi(x_0 + x_1) = 0 && \text{By Eq. 3.10} \\
&\iff r_0(\phi+1) + \phi(x_0 + x_1) = 0 \\
&\iff r_0(\phi+1) + \phi(x_0 + x_1) = 0 \\
&\iff False
\end{aligned}
$$

Since $r_0, \phi, x_0, x_1 \in \mathbb{R}_{>0}$

□

From this lemma, we know that the value that brings the AMM to the equilibrium is unique. It is now natural to ask if such value is also the optimal value that maximizes the user's gain like it was for the previous model. However, this is not the case.

Example 3.5.3. Let $\Gamma = \mathsf{A}[30 : \tau_0, 20 : \tau_1] \mid \{10 : \tau_0, 30 : \tau_1\}$. Now suppose that the prices of τ_0 and τ_1 are $P(\tau_0) = 4$ and $P(\tau_1) = 5$. Let $\mathsf{T}(x) = \mathsf{A} : \mathsf{swap}(x, \tau_0, \tau_1)$; in particular, we pick $\mathsf{T}(x_0)$ where x_0 is the one defined in Eq 3.6 and suppose that SX_ϕ is the constant product swap rate and that $\phi = 0.9$; this time we exaggerate with the fee just to show more evident results.

By computing the gain of A by firing $\mathsf{T}(x_0)$ we get:

$$G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) \approx 31.098 \tag{3.12}$$

In the model defined in [8], this would have been the maximum gain that A could get by firing a swap in the state Γ. This, however, is not the case in our model. Figure Figure 3.2 shows the plot of $G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0))$. Clearly we can see that this function is concave, i.e. it has a critical point which is also the absolute maximum for the function. However, looking closely and zooming in, we can see in Figure Figure 3.3 that the gain at x_0 (in blue) is not exactly at the top of the curve. In fact, if we try computing the gain with a slightly bigger $x_1 = x_0 + 0.3$ we get:

$$G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_1)) \approx 31.138$$

which is indeed bigger than the one in Eq. 3.12.

Hence, we know that this value is not the optimal one. This is, however, very close to it, and it is still worth comparing the gain of this value to the one of every other value, which is done by the theorem below.

Theorem 3.5.4 ▶ Equilibrium value vs gain

Let $\Gamma = \mathsf{A}[\sigma_\mathsf{A}] \mid \{r_0 : \tau_0, r_1 : \tau_1\} \mid \Delta$ be such that $S_\Gamma\{\tau_0, \tau_1\} > 0$ and $\sigma_\mathsf{A}\{\tau_0, \tau_1\} < S_\Gamma\{\tau_0, \tau_1\}$ and let $\mathsf{T}(x) = \mathsf{A} : \mathsf{swap}(x, \tau_0, \tau_1)$. Let x_0 be the one defined in Eq. 3.6 and assume that $\mathsf{T}(x_0)$ is enabled in Γ.
If SX_ϕ is the constant product swap rate and $\phi < 1$, then:

$$\forall x < x_0 \ : \ G_\mathsf{A}(\Gamma, \mathsf{T}(x_0)) \ > \ G_\mathsf{A}(\Gamma, \mathsf{T}(x))$$

and

$$\forall x > x_0 \ : \ (G_\mathsf{A}(\Gamma, \mathsf{T}(x_0)) \ > \ G_\mathsf{A}(\Gamma, \mathsf{T}(x)) \iff x > \frac{x_0}{\phi})$$

assuming that $\mathsf{T}(x)$ is also enabled in Γ.

Proof.

- Case $x < x_0$:

  Let $x_1 > 0$ be such that $x_0 = x + x_1$. Since SX_ϕ is output bounded, then $\mathsf{T}(x_0)$, $\mathsf{T}(x)$ and $\mathsf{T}(x_1)$ are all enabled, in particular :

  $$\Gamma \xrightarrow{\mathsf{T}(x_0)} \Gamma'$$
  $$\Gamma \xrightarrow{\mathsf{T}(x)} \Gamma_1 \xrightarrow{\mathsf{T}(x_1)} \Gamma_2$$

  By Theorem 3.2.8 we know that:

  $$G_\mathsf{A}(\Gamma, \mathsf{T}(x_0)) \ = \ G_\mathsf{A}(\Gamma, \mathsf{T}(x)) + G_\mathsf{A}(\Gamma_1, \mathsf{T}(x_1)) + \varepsilon_\phi$$

  Since we know that $\varepsilon_\phi > 0$ we just have to prove that $G_\mathsf{A}(\Gamma_1, \mathsf{T}(x_1)) > 0$. To do that, we first prove that $SX_\phi(x_1, r_0 + x, r_1 - \alpha x) > X(\tau_0, \tau_1)$ where $\alpha = SX_\phi(x, r_0, r_1)$:

  $$\begin{aligned} X(\tau_0, \tau_1) &< X_{\Gamma_2}(\tau_0, \tau_1) && \text{by 3.4.2} \\ &< SX_\phi(x_1, r_0 + x, r_1 - \alpha x) && \text{by 3.4.1} \end{aligned}$$

  So, by applying Lemma 3.1.4 we can finish the proof.

- Case $x > x_0$: Let $x_1 > 0$ be such that $x = x_0 + x_1$. Since SX_ϕ is output bounded, then $\mathsf{T}(x_0)$, $\mathsf{T}(x)$ and $\mathsf{T}(x_1)$ are all enabled, in particular :

  $$\Gamma \xrightarrow{\mathsf{T}(x)} \Gamma_1$$
  $$\Gamma \xrightarrow{\mathsf{T}(x_0)} \Gamma' \xrightarrow{\mathsf{T}(x_1)} \Gamma_2$$

  By Theorem 3.2.8 we know that:

  $$G_\mathsf{A}(\Gamma, \mathsf{T}(x)) \ = \ G_\mathsf{A}(\Gamma, \mathsf{T}(x_0)) + G_\mathsf{A}(\Gamma', \mathsf{T}(x_1)) + \varepsilon_\phi$$

To finish our proof, we need to prove that $G_{\mathsf{A}}(\Gamma', \mathsf{T}(x_1)) + \varepsilon_\phi < 0$. Since we know that $\varepsilon_\phi > 0$, we have first to see if $G_{\mathsf{A}}(\Gamma', \mathsf{T}(x_1)) < 0$, otherwise it is always false. So, we want to show that $SX_\phi(x_1, r_0 + x, r_1 - \alpha x) < X(\tau_0, \tau_1)$ where $\alpha = SX_\phi(x_0, r_0, r_1)$:

$$\begin{aligned} X(\tau_0, \tau_1) &= X_{\Gamma'}(\tau_0, \tau_1) \\ &< SX_\phi(x_1, r_0 + x_0, r_1 - \alpha x_0) \qquad \text{by 3.4.1} \end{aligned}$$

So, by applying Lemma 3.1.4 we know that $G_{\mathsf{A}}(\Gamma', \mathsf{T}(x_1)) < 0$, hence we can continue the proof.

Before proceeding with the proof, we point out the values of some terms that we use during the proof:

$$\eta = 1 - \frac{\sigma_{\mathsf{A}}\{\tau_0, \tau_1\}}{S_\Gamma\{\tau_0, \tau_1\}}$$

$$\begin{aligned} x_1\beta &= x_1 \cdot \frac{\phi\left(r_1 - \frac{\phi r_1}{r_0 + \phi x_0} \cdot x_0\right)}{r_0 + x_0 + \phi x_1} = x_1 \cdot \frac{\phi\left(\frac{r_1(r_0 + \phi x_0) - \phi r_1 x_0}{r_0 + \phi x_0}\right)}{r_0 + x_0 + \phi x_1} \\ &= x_1 \cdot \frac{\phi\left(\frac{r_1 r_0 + \phi r_1 x_0 - \phi r_1 x_0}{r_0 + \phi x_0}\right)}{r_0 + x_0 + \phi x_1} = \frac{\phi r_0 r_1 x_1}{(r_0 + x_0 + \phi x_1)(r_0 + \phi x_0)} \end{aligned}$$

$$\begin{aligned} (\alpha x_0 + \beta x_1) &= \frac{\phi r_1 x_0}{r_0 + \phi x_0} + \frac{\phi r_0 r_1 x_1}{(r_0 + x_0 + \phi x_1)(r_0 + \phi x_0)} = \frac{\phi r_1 x_0 (r_0 + x_0 + \phi x_1) + \phi r_0 r_1 x_1}{(r_0 + x_0 + \phi x_1)(r_0 + \phi x_0)} \\ &= \frac{\phi r_0 r_1 x_0 + \phi r_1 x_0^2 + \phi^2 r_1 x_0 x_1 + \phi r_0 r_1 x_1}{(r_0 + x_0 + \phi x_1)(r_0 + \phi x_0)} = \frac{\phi r_1 (r_0 x_0 + x_0^2 + \phi x_0 x_1 + r_0 x_1)}{(r_0 + x_0 + \phi x_1)(r_0 + \phi x_0)} \end{aligned}$$

$$\begin{aligned} (z - 1) &= \frac{(x_0 + x_1)(r_0 + \phi x_0)(r_0 + x_0 + \phi x_1)}{(r_0 + \phi x_0 + \phi x_1)(x_0^2 + r_0 x_0 + \phi x_0 x_1 + r_0 x_1)} - 1 \\ &= \frac{(r_0 x_0 + \phi x_0^2 + r_0 x_1 + \phi x_0 x_1)(r_0 + x_0 + \phi x_1)}{(r_0 + \phi x_0 + \phi x_1)(x_0^2 + r_0 x_0 + \phi x_0 x_1 + r_0 x_1)} \\ &\quad - \frac{(r_0 + \phi x_0 + \phi x_1)(r_0 x_0 + x_0^2 + r_0 x_1 + \phi x_0 x_1)}{(r_0 + \phi x_0 + \phi x_1)(x_0^2 + r_0 x_0 + \phi x_0 x_1 + r_0 x_1)} \\ &= -\frac{r_0 x_0 x_1 (\phi - 1)}{(r_0 + \phi x_0 + \phi x_1)(x_0^2 + r_0 x_0 + \phi x_0 x_1 + r_0 x_1)} \end{aligned}$$

$$\begin{aligned} &x_1\beta + (z - 1)(\alpha x_0 + \beta x_1) \\ &\quad = \frac{\phi r_0 r_1 x_1}{(r_0 + x_0 + \phi x_1)(r_0 + \phi x_0)} - \frac{r_0 r_1 x_0 x_1 (\phi - 1)\phi}{(r_0 + \phi x_0 + \phi x_1)(r_0 + \phi x_0)(r_0 + x_0 + \phi x_1)} \\ &\quad = \frac{\phi r_0 r_1 x_1 (r_0 + \phi x_0 + \phi x_1) - r_0 r_1 x_0 x_1 (\phi - 1)\phi}{(r_0 + \phi x_0 + \phi x_1)(r_0 + \phi x_0)(r_0 + x_0 + \phi x_1)} \\ &\quad = \frac{\phi r_0 r_1 x_1 ((r_0 + \phi x_0 + \phi x_1) - x_0 (\phi - 1))}{(r_0 + \phi x_0 + \phi x_1)(r_0 + \phi x_0)(r_0 + x_0 + \phi x_1)} \end{aligned}$$

$$
\begin{aligned}
&= \frac{\phi r_0 r_1 x_1 (r_0 + x_0 + \phi x_1)}{(r_0 + \phi x_0 + \phi x_1)(r_0 + \phi x_0)(r_0 + x_0 + \phi x_1)} \\
&= \frac{\phi r_0 r_1 x_1}{(r_0 + \phi x_0 + \phi x_1)(r_0 + \phi x_0)}
\end{aligned} \tag{3.13}
$$

$$
\begin{aligned}
X_{\Gamma'}(\tau_0, \tau_1) &= \lim_{z \to 0} SX_\phi(z, r_0 + x_0, r_1 - \alpha x_0) \\
&= \frac{\phi(r_1 - \alpha x_0)}{r_0 + x_0} = \frac{\phi\left(\frac{r_1(r_0 + \phi x_0) - \phi r_1 x_0}{r_0 + \phi x_0}\right)}{r_0 + x_0} = \frac{\phi r_0 r_1}{(r_0 + \phi x_0)(r_0 + x_0)}
\end{aligned}
$$

With these values in mind, we can continue the proof:

$$
\begin{aligned}
& G_{\mathsf{A}}(\Gamma', \mathsf{T}(x_1)) + \varepsilon_\phi < 0 \\
\iff & \eta(x_1(\beta P(\tau_1) - P(\tau_0)) + P(\tau_1)(z-1)(\alpha x_0 + \beta x_1)) < 0 \\
\iff & x_1(\beta P(\tau_1) - P(\tau_0)) + P(\tau_1)(z-1)(\alpha x_0 + \beta x_1) < 0 \\
\iff & x_1 \beta P(\tau_1) - x_1 P(\tau_0) + P(\tau_1)(z-1)(\alpha x_0 + \beta x_1) < 0 \\
\iff & P(\tau_1)(x_1 \beta + (z-1)(\alpha x_0 + \beta x_1)) < x_1 P(\tau_0) \\
\iff & \frac{x_1 \beta + (z-1)(\alpha x_0 + \beta x_1)}{x_1} < \frac{P(\tau_0)}{P(\tau_1)} && \text{by } x_1 > 0 \wedge P(\tau_1) > 0 \\
\iff & \frac{x_1 \beta + (z-1)(\alpha x_0 + \beta x_1)}{x_1} < X_{\Gamma'}(\tau_0, \tau_1) \\
\iff & \frac{\frac{\phi r_0 r_1 x_1}{(r_0 + \phi x_0 + \phi x_1)(r_0 + \phi x_0)}}{x_1} < X_{\Gamma'}(\tau_0, \tau_1) \\
\iff & \frac{\phi r_0 r_1}{(r_0 + \phi x_0 + \phi x_1)(r_0 + \phi x_0)} < \frac{\phi r_0 r_1}{(r_0 + \phi x_0)(r_0 + x_0)} \\
\iff & \frac{\phi r_0 r_1 (r_0 + x_0) - \phi r_0 r_1 (r_0 + \phi x_0 + \phi x_1)}{(r_0 + \phi x_0 + \phi x_1)(r_0 + \phi x_0)(r_0 + x_0)} < 0 \\
\iff & \phi r_0 r_1 (r_0 + x_0) - \phi r_0 r_1 (r_0 + \phi x_0 + \phi x_1) < 0 && \text{by Den. } > 0 \\
\iff & \phi r_0 r_1 (r_0 + x_0 - r_0 - \phi x_0 - \phi x_1) < 0 \\
\iff & x_0 - \phi x_0 - \phi x_1 < 0 && \text{by } \phi r_0 r_1 < 0 \\
\iff & \phi(x_0 + x_1) - x_0 > 0 \\
\iff & \phi > \frac{x_0}{x_0 + x_1} \\
\iff & x_1 > \frac{x_0(1-\phi)}{\phi}
\end{aligned}
$$

So, this is only true when:

$$
\begin{aligned}
\iff & x_1 > \frac{x_0(1-\phi)}{\phi} \\
\iff & x - x_0 > \frac{x_0(1-\phi)}{\phi} \\
\iff & x > \frac{x_0(1-\phi) + \phi x_0}{\phi}
\end{aligned}
$$

$$\iff x > \frac{x_0}{\phi}$$

□

This is what we expected from Example 3.5.3 looking at the Figures 3.2 and 3.3, i.e. there is a small interval after x_0 where the gain that A could get is better than the one obtained by swapping x_0 tokens. We know that this interval is between x_0 and $\frac{x_0}{\phi}$, hence, the only thing left to do is finding such a value that maximizes the gain.

LEAN

Theorem 3.5.5 ▶ Max Gain Value

Let $\Gamma = \mathsf{A}[\sigma_\mathsf{A}] \mid \{r_0 : \tau_0, r_1 : \tau_1\}$ be such that $S_\Gamma\{\tau_0, \tau_1\} > 0$ and $\sigma_\mathsf{A}\{\tau_0, \tau_1\} < S_\Gamma\{\tau_0, \tau_1\}$, and let $\mathsf{T}(x) = \mathsf{A} : \mathsf{swap}(x, \tau_0, \tau_1)$. Let x_0 be such that:

$$X_{\Gamma'}(\tau_0, \tau_1) = X(\tau_0, \tau_1) \qquad \text{where } \Gamma \xrightarrow{\mathsf{T}(x_0)} \Gamma' \tag{3.14}$$

Let x_{max} be

$$x_{max} = x_0 + \frac{-\sqrt{P(\tau_0)}r_0 - \sqrt{P(\tau_0)}\phi x_0 + \sqrt{P(\tau_1)\phi r_0 r_1}}{\sqrt{P(\tau_0)}\phi} \tag{3.15}$$

If SX_ϕ is the constant product swap rate function, $\mathsf{T}(x_{max})$ is enabled in Γ and $\phi < 1$, then:

$$\forall x \neq x_{max} \ : \ G_\mathsf{A}(\Gamma, \mathsf{T}(x_{max})) \ > \ G_\mathsf{A}(\Gamma, \mathsf{T}(x))$$

Proof. First, we show that x_{max} is positive only if $\phi < 1$:

$$\begin{aligned}
& x_0 + \frac{-\sqrt{P(\tau_0)}r_0 - \sqrt{P(\tau_0)}\phi x_0 + \sqrt{P(\tau_1)\phi r_0 r_1}}{\sqrt{P(\tau_0)}\phi} > 0 \\
\iff & \frac{-\sqrt{P(\tau_0)}r_0 - \sqrt{P(\tau_0)}\phi x_0 + \sqrt{P(\tau_1)\phi r_0 r_1}}{\sqrt{P(\tau_0)}\phi} > 0 \\
\iff & -\sqrt{P(\tau_0)}r_0 - \sqrt{P(\tau_0)}\phi x_0 + \sqrt{P(\tau_1)\phi r_0 r_1} > 0 \\
\iff & -\sqrt{P(\tau_0)}(r_0 + \phi x_0) > \sqrt{P(\tau_1)\phi r_0 r_1} \\
\iff & P(\tau_0)(r_0 + \phi x_0)^2 < P(\tau_1)\phi r_0 r_1 \\
\iff & \frac{P(\tau_0)}{P(\tau_1)} < \frac{\phi r_0 r_1}{(r_0 + \phi x_0)^2} \\
\iff & \frac{\phi(r_1 - \alpha x_0)}{r_0 + x_0} < \frac{\phi r_0 r_1}{(r_0 + \phi x_0)^2} \\
\iff & \frac{\phi r_0 r_1}{(r_0 + x_0)(r_0 \phi x_0)} < \frac{\phi r_0 r_1}{(r_0 + \phi x_0)^2} \\
\iff & \frac{1}{r_0 + x_0} < \frac{1}{(r_0 + \phi x_0)} \\
\iff & r_0 + x_0 > r_0 + \phi x_0 \\
\iff & x_0 > \phi x_0
\end{aligned}$$

$$\iff 1 > \phi$$

Let $x_1 > 0$ be such that $x_{max} = x_0 + x_1$. Then, as we stated in the proof of Theorem 3.5.4, by Theorem 3.2.8 we know that:

$$G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_{max})) \;=\; G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + G_{\mathsf{A}}(\Gamma', \mathsf{T}(x_1)) + \varepsilon_\phi$$

We have proven in Theorem 3.5.4 that for $x < \frac{x_0}{\phi}$, $G_{\mathsf{A}}(\Gamma', \mathsf{T}(x_1)) + \varepsilon_\phi > 0$. Since x_0 is fixed, we want to find the maximum of the function:

$$f(x_1) = G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + G_{\mathsf{A}}(\Gamma', \mathsf{T}(x_1)) + \varepsilon_\phi \tag{3.16}$$

To find the maximum, first we look at the function values at the extremes of its domain:

$$\begin{aligned}
\lim_{x_1 \to 0} f(x_1) &= \lim_{x_1 \to 0} G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + G_{\mathsf{A}}(\Gamma', \mathsf{T}(x_1)) + \varepsilon_\phi \\
&= \lim_{x_1 \to 0} G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + \eta(x_1 \beta P(\tau_1) - x_1 P(\tau_0) + P(\tau_1)(z-1)(\alpha x_0 + \beta x_1)) \\
&= \lim_{x_1 \to 0} G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + \eta(-x_1 P(\tau_0) + P(\tau_1)((z-1)(\alpha x_0 + \beta x_1) + \beta x_1)) \\
&= \lim_{x_1 \to 0} G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + \eta\left(-x_1 P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1 x_1}{(r_0 + \phi x_0)(r_0 + \phi x_0 + \phi x_1)}\right) && \text{(By 3.13)} \\
&= G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + \eta\left(-0 + \frac{0}{(r_0 + \phi x_0)(r_0 + \phi x_0)}\right) \\
&= G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0))
\end{aligned}$$

$$\begin{aligned}
\lim_{x_1 \to \infty} f(x_1) &= \lim_{x_1 \to \infty} G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + G_{\mathsf{A}}(\Gamma', \mathsf{T}(x_1)) + \varepsilon_\phi \\
&= \lim_{x_1 \to \infty} G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + \eta\left(-x_1 P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1 x_1}{(r_0 + \phi x_0)(r_0 + \phi x_0 + \phi x_1)}\right) \\
&= \lim_{x_1 \to \infty} G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + \eta\left(-x_1 P(\tau_0) + \frac{x_1 P(\tau_1)\phi r_0 r_1}{x_1(r_0 + \phi x_0)(\frac{r_0}{x_1} + \frac{\phi x_0}{x_1} + \phi)}\right) \\
&= G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + \eta\left(-\infty P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1}{(r_0 + \phi x_0)(\frac{r_0}{x_1} + \frac{\phi x_0}{x_1} + \phi)}\right) \\
&= -\infty
\end{aligned}$$

Now, we have to study the critical points of the function. To do that, we first have to compute the first derivative of $f(x_1)$:

$$\begin{aligned}
f'(x_1) &= \Big[- x_1 P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1 x_1}{(r_0 + \phi x_0)(r_0 + \phi x_0 + \phi x_1)} \Big]' \\
&= - P(\tau_0) + \Big[\frac{P(\tau_1)\phi r_0 r_1 x_1}{(r_0 + \phi x_0)(r_0 + \phi x_0 + \phi x_1)} \Big]'
\end{aligned}$$

$$
\begin{aligned}
&= -P(\tau_0) + \frac{[P(\tau_1)\phi r_0 r_1 x_1]'(r_0+\phi x_0)(r_0+\phi x_0+\phi x_1)}{((r_0+\phi x_0)(r_0+\phi x_0+\phi x_1))^2} \\
&\quad - \frac{(P(\tau_1)\phi r_0 r_1 x_1)[(r_0+\phi x_0)(r_0+\phi x_0+\phi x_1)]'}{((r_0+\phi x_0)(r_0+\phi x_0+\phi x_1))^2} \\
&= -P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1 (r_0+\phi x_0)(r_0+\phi x_0+\phi x_1)}{((r_0+\phi x_0)(r_0+\phi x_0+\phi x_1))^2} \\
&\quad - \frac{(P(\tau_1)\phi r_0 r_1 x_1)\phi(r_0+\phi x_0)}{((r_0+\phi x_0)(r_0+\phi x_0+\phi x_1))^2} \\
&= -P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1 \Big((r_0+\phi x_0)(r_0+\phi x_0+\phi x_1) - \phi x_1(r_0+\phi x_0)\Big)}{(r_0+\phi x_0)^2(r_0+\phi x_0+\phi x_1)^2} \\
&= -P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1 (r_0+\phi x_0)\Big(r_0+\phi x_0+\phi x_1-\phi x_1\Big)}{(r_0+\phi x_0)^2(r_0+\phi x_0+\phi x_1)^2} \\
&= -P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1 (r_0+\phi x_0)(r_0+\phi x_0)}{(r_0+\phi x_0)^2(r_0+\phi x_0+\phi x_1)^2} \\
&= -P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1}{(r_0+\phi x_0+\phi x_1)^2}
\end{aligned}
$$

Now, we want to prove that

$$
x' = \frac{-\sqrt{P(\tau_0)}r_0 - \sqrt{P(\tau_0)}\phi x_0 + \sqrt{P(\tau_1)\phi r_0 r_1}}{\sqrt{P(\tau_0)}\phi} \tag{3.17}
$$

is a critical point of $f(x_1)$, i.e. $f'(x') = 0$. First, let's rewrite:

$$
\begin{aligned}
\phi x' &= \phi \cdot \frac{-\sqrt{P(\tau_0)}r_0 - \sqrt{P(\tau_0)}\phi x_0 + \sqrt{P(\tau_1)\phi r_0 r_1}}{\sqrt{P(\tau_0)}\phi} \\
&= \frac{-\sqrt{P(\tau_0)}r_0 - \sqrt{P(\tau_0)}\phi x_0 + \sqrt{P(\tau_1)\phi r_0 r_1}}{\sqrt{P(\tau_0)}} \\
&= \frac{-\sqrt{P(\tau_0)}(r_0+\phi x_0)}{\sqrt{P(\tau_0)}} + \frac{\sqrt{P(\tau_1)\phi r_0 r_1}}{\sqrt{P(\tau_0)}} \\
&= -(r_0+\phi x_0) + \frac{\sqrt{P(\tau_1)\phi r_0 r_1}}{\sqrt{P(\tau_0)}}
\end{aligned} \tag{3.18}
$$

Finally,

$$
\begin{aligned}
f'(x') &= -P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1}{(r_0+\phi x_0+\phi x')^2} \\
&= -P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1}{(r_0+\phi x_0-(r_0+\phi x_0)+\frac{\sqrt{P(\tau_1)\phi r_0 r_1}}{\sqrt{P(\tau_0)}})^2} \\
&= -P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1}{(\frac{\sqrt{P(\tau_1)\phi r_0 r_1}}{\sqrt{P(\tau_0)}})^2}
\end{aligned}
$$

$$
\begin{aligned}
&= -P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1}{\frac{P(\tau_1)\phi r_0 r_1}{P(\tau_0)}} \\
&= -P(\tau_0) + P(\tau_0) = 0
\end{aligned}
$$

Now, since we know that this is either a local maximum or a local minimum, we only have to study the sign of the second derivative of $f(x_1)$ to determine which is the case. Let's compute it first:

$$
\begin{aligned}
f''(x_1) &= \Big[-P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1}{(r_0 + \phi x_0 + \phi x_1)^2} \Big]' \\
&= \Big[\frac{P(\tau_1)\phi r_0 r_1}{(r_0 + \phi x_0 + \phi x_1)^2} \Big]' \\
&= P(\tau_1)\phi r_0 r_1 \cdot \Big[(r_0 + \phi x_0 + \phi x_1)^{-2} \Big]' \\
&= P(\tau_1)\phi r_0 r_1 \cdot (-2) * (r_0 + \phi x_0 + \phi x_1)^{-3} * [(r_0 + \phi x_0 + \phi x_1)]' \\
&= P(\tau_1)\phi r_0 r_1 \cdot (-2) * (r_0 + \phi x_0 + \phi x_1)^{-3} * \phi \\
&= -2P(\tau_1)\phi^2 r_0 r_1 * (r_0 + \phi x_0 + \phi x_1)^{-3}
\end{aligned}
$$

Since $x_1 > 0$ and all the other constant are also strictly positive, then it is easy to see that this second derivative is always negative for $x_1 > 0$, hence in the function's domain, the function is concave. This is enough to prove that x' is a global maximum over the functions domain [15]

Unfortunately, due to the limitation of the Lean 4 implementation of the model, it is not possible to conduct the proof this way in the theorem prover. Therefore, we present also the proof outlined in the implementation.

Again, consider the function

$$
\begin{aligned}
f(x_1) &= G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + G_{\mathsf{A}}(\Gamma', \mathsf{T}(x_1)) + \varepsilon_\phi \\
&= G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + \eta(x_1 \beta P(\tau_1) - x_1 P(\tau_0) + P(\tau_1)(z-1)(\alpha x_0 + \beta x_1)) \\
&= G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + \eta(-x_1 P(\tau_0) + P(\tau_1)((z-1)(\alpha x_0 + \beta x_1) + \beta x_1)) \\
&= G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + \eta \left(-x_1 P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1 x_1}{(r_0 + \phi x_0)(r_0 + \phi x_0 + \phi x_1)} \right) \qquad \text{(By 3.13)}
\end{aligned}
$$

We want to prove that

$$
x' = \frac{-\sqrt{P(\tau_0)} r_0 - \sqrt{P(\tau_0)}\phi x_0 + \sqrt{P(\tau_1)\phi r_0 r_1}}{\sqrt{P(\tau_0)}\phi} \tag{3.19}
$$

Is the global maximum of $f(x_1)$, i.e.

$$
\forall x \neq x', f(x') > f(x)
$$

First, let's rewrite the inequality:

$$
\begin{aligned}
& f(x') > f(x) \\
\iff & G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + \eta\left(-x'P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1 x'}{(r_0+\phi x_0)(r_0+\phi x_0+\phi x')}\right) > \\
& G_{\mathsf{A}}(\Gamma, \mathsf{T}(x_0)) + \eta\left(-xP(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1 x}{(r_0+\phi x_0)(r_0+\phi x_0+\phi x)}\right) \\
\iff & -x'P(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1 x'}{(r_0+\phi x_0)(r_0+\phi x_0+\phi x')} > \\
& -xP(\tau_0) + \frac{P(\tau_1)\phi r_0 r_1 x}{(r_0+\phi x_0)(r_0+\phi x_0+\phi x)} \\
\iff & \frac{P(\tau_1)\phi r_0 r_1 x'}{(r_0+\phi x_0)(r_0+\phi x_0+\phi x')} - x'P(\tau_0) - \left(\frac{P(\tau_1)\phi r_0 r_1 x}{(r_0+\phi x_0)(r_0+\phi x_0+\phi x)} - xP(\tau_0)\right) > 0
\end{aligned}
$$

Now let $A = r_0 + \phi x_0$. Then, the inequality becomes

$$
\begin{aligned}
& \frac{P(\tau_1)\phi r_0 r_1 x'}{(r_0+\phi x_0)(r_0+\phi x_0+\phi x')} - x'P(\tau_0) - \left(\frac{P(\tau_1)\phi r_0 r_1 x}{(r_0+\phi x_0)(r_0+\phi x_0+\phi x)} - xP(\tau_0)\right) > 0 \\
\iff & \frac{P(\tau_1)\phi r_0 r_1 x'}{A(A+\phi x')} - x'P(\tau_0) - \left(\frac{P(\tau_1)\phi r_0 r_1 x}{A(A+\phi x)} - xP(\tau_0)\right) > 0 \\
\iff & \frac{P(\tau_1)\phi r_0 r_1 x' - x'P(\tau_0)A(A+\phi x')}{A(A+\phi x')} - \left(\frac{P(\tau_1)\phi r_0 r_1 x - xP(\tau_0)A(A+\phi x)}{A(A+\phi x)}\right) > 0
\end{aligned}
$$

We define the terms $D(x) = A(A+\phi x)$ and $N(x) = P(\tau_1)\phi r_0 r_1 x - P(\tau_0)xA(A+\phi x)$ to simplify the expression above:

$$
\begin{aligned}
& \frac{P(\tau_1)\phi r_0 r_1 x' - x'P(\tau_0)A(A+\phi x')}{A(A+\phi x')} - \left(\frac{P(\tau_1)\phi r_0 r_1 x - xP(\tau_0)A(A+\phi x)}{A(A+\phi x)}\right) > 0 \\
\iff & \frac{N(x')}{D(x')} - \frac{N(x)}{D(x)} > 0 \\
\iff & N(x')D(x) - N(x)D(x') > 0 \\
\iff & \Big(P(\tau_1)\phi r_0 r_1 x' - x'P(\tau_0)A(A+\phi x')\Big)(A(A+\phi x)) \\
& - \Big(P(\tau_1)\phi r_0 r_1 x - xP(\tau_0)A(A+\phi x)\Big)(A(A+\phi x')) > 0 \\
\iff & A\Big[\Big(P(\tau_1)\phi r_0 r_1 x' - x'P(\tau_0)A(A+\phi x')\Big)(A+\phi x) \\
& - \Big(P(\tau_1)\phi r_0 r_1 x - xP(\tau_0)A(A+\phi x)\Big)(A+\phi x')\Big] > 0 \qquad (3.20)
\end{aligned}
$$

Let this term be

$$\Delta(x) = A\Delta^*(x)$$

Now, observe that

$$\Delta^*(x) = \Big(P(\tau_1)\phi r_0 r_1 x' - x'P(\tau_0)A(A+\phi x')\Big)(A+\phi x)$$

$$
\begin{aligned}
&- \Big(P(\tau_1)\phi r_0 r_1 x - xP(\tau_0)A(A+\phi x)\Big)(A+\phi x') \\
&= P(\tau_1)\phi r_0 r_1 \Big(x'(A+\phi x) - x(A+\phi x')\Big) \\
&- P(\tau_0)A\Big(x'(A+\phi x')(A+\phi x) - x(A+\phi x)(A+\phi x')\Big) \\
&= P(\tau_1)\phi r_0 r_1 \Big(x'(A+\phi x) - x(A+\phi x')\Big) \\
&- P(\tau_0)A(x'-x)(A+\phi x')(A+\phi x) \\
&= P(\tau_1)\phi r_0 r_1 A(x'-x) \\
&- P(\tau_0)A(x'-x)(A+\phi x')(A+\phi x) \\
&= A(x'-x)\Big[P(\tau_1)\phi r_0 r_1 - P(\tau_0)(A+\phi x')(A+\phi x)\Big]
\end{aligned}
$$

Hence,

$$
\Delta(x) = A^2(x'-x)\Big[P(\tau_1)\phi r_0 r_1 - P(\tau_0)(A+\phi x')(A+\phi x)\Big]
$$

Now, recall the definition 3.19. We can find a useful equality condition starting from this:

$$
\begin{aligned}
x' &= \frac{-\sqrt{P(\tau_0)}r_0 - \sqrt{P(\tau_0)}\phi x_0 + \sqrt{P(\tau_1)\phi r_0 r_1}}{\sqrt{P(\tau_0)}\phi} \\
\iff \sqrt{P(\tau_0)}\phi x' &= -\sqrt{P(\tau_0)}(r_0 + \phi x_0) + \sqrt{P(\tau_1)\phi r_0 r_1} \\
\iff \sqrt{P(\tau_0)}\phi x' + \sqrt{P(\tau_0)}A &= \sqrt{P(\tau_1)\phi r_0 r_1} \\
\iff \sqrt{P(\tau_0)}(A+\phi x') &= \sqrt{P(\tau_1)\phi r_0 r_1} \\
\iff P(\tau_0)(A+\phi x')^2 &= P(\tau_1)\phi r_0 r_1
\end{aligned}
$$

Now recall the definition of $\Delta(x)$. With the above equality we get:

$$
\begin{aligned}
\Delta(x) &= A^2(x'-x)\Big[P(\tau_1)\phi r_0 r_1 - P(\tau_0)(A+\phi x')(A+\phi x)\Big] \\
&= A^2(x'-x)\Big[P(\tau_0)(A+\phi x')^2 - P(\tau_0)(A+\phi x')(A+\phi x)\Big] \\
&= A^2(x'-x)\Big[P(\tau_0)(A+\phi x')((A+\phi x') - (A+\phi x))\Big] \\
&= A^2(x'-x)\Big[P(\tau_0)(A+\phi x')\phi(x'-x)\Big] \\
&= A^2 P(\tau_0)(A+\phi x')\phi(x'-x)^2
\end{aligned}
$$

Finally, combining this with (3.20) yields:

$$
\begin{aligned}
& f(x') > f(x) \\
\iff & \Delta(x) > 0 \\
\iff & A^2 P(\tau_0)(A+\phi x')\phi(x'-x)^2 > 0 \\
\iff & (x'-x)^2 > 0
\end{aligned}
$$

$$\iff x' - x \neq 0$$
$$\iff x' \neq x$$

Since $A > 0$, $P(\tau_0) > 0$, $\phi > 0$, $x' > 0$ and $x' \neq x$ by hypothesis. □

Finally, it is also worth exploring if this is the only value that maximizes the gain, and this can be easily proved by contradiction, stated in the lemma below.

Lemma 3.5.6 ▶ Optimal swap value uniqueness

Let $\Gamma = \mathsf{A}[\sigma_\mathsf{A}] \mid \{r_0 : \tau_0, r_1 : \tau_1\}$ be such that $S_\Gamma\{\tau_0, \tau_1\} > 0$ and $\sigma_\mathsf{A}\{\tau_0, \tau_1\} < S_\Gamma\{\tau_0, \tau_1\}$, and $\mathsf{T}(x) = \mathsf{A} : \mathsf{swap}(x, \tau_0, \tau_1)$. If SX_ϕ is the constant product swap rate, $\phi < 1$ and

$$X(\tau_0, \tau_1) = X_{\Gamma'}(\tau_0, \tau_1) \qquad \text{where } \Gamma \xrightarrow{\mathsf{T}(x_0)} \Gamma'$$

then

$$\neg\exists\, x_1 \in \mathbb{R} > 0 \;:\; x_1 \neq x_{max} \;\wedge\; \forall x \neq x_1 \;:\; G_\mathsf{A}(\Gamma, \mathsf{T}(x_1)) \;>\; G_\mathsf{A}(\Gamma, \mathsf{T}(x))$$
$$\text{where } \Gamma \xrightarrow{\mathsf{T}(x_1)} \Gamma' \text{ and } \Gamma \xrightarrow{\mathsf{T}(x)} \Gamma''$$

i.e. x_0 is unique.

Proof. First, from Theorem 3.5.5 we know that

$$\forall x \neq x_{max} \;:\; G_\mathsf{A}(\Gamma, \mathsf{T}(x_{max})) \;>\; G_\mathsf{A}(\Gamma, \mathsf{T}(x)) \tag{3.21}$$

Now, by contradiction, assume that such x_1 exists. Hence, we know that

$$x_1 \neq x_{max} \;\wedge\; \forall x \neq x_1 \;:\; G_\mathsf{A}(\Gamma, \mathsf{T}(x_1)) \;>\; G_\mathsf{A}(\Gamma, \mathsf{T}(x)) \tag{3.22}$$

1. Case $x_1 \neq x_{max}$: Instantiating 3.22 we get

$$G_\mathsf{A}(\Gamma, \mathsf{T}(x_1)) \;>\; G_\mathsf{A}(\Gamma, \mathsf{T}(x_{max})) \tag{3.23}$$

   And instantiating 3.21 we get:

$$G_\mathsf{A}(\Gamma, \mathsf{T}(x_{max})) \;>\; G_\mathsf{A}(\Gamma, \mathsf{T}(x_1)) \tag{3.24}$$

   Hence, combining 3.23 and 3.24 we reach a contradiction.

2. Case $x_1 = x_{max}$: Trivial with 3.22.

□

Chapter 4

Conclusions

In this thesis, we have presented a rigorous and machine-checked analysis of Automated Market Makers (AMMs) with trading fees, built on the abstract operational model presented in [8] and its Lean 4 formalization in [5]. This work has extended these foundations with three main contributions:

1. **Formal modeling of trading fees**
   We introduced the trading fee ϕ as a parameter of the constant product swap rate:

   $$SX_{\phi}(x, r_0, r_1) = \frac{\phi r_1}{r_0 + \phi \cdot x}$$

   This enriches the pure constant-product model with a fee mechanism that brings the overall model closer to the Uniswap v2 implementation, allowing one to reason over the impact of fees in the AMM's reserves, swap output token amount and user's wealth. In Equation 2.1 we have also shown that, the product of the reserves of an AMM $r_0 r_1$ does not always remain constant after a swap, i.e.

   $$(r_0 + x)(r_1 - y) \geq r_0 r_1 \qquad \text{for } \phi \leq 1$$

   In particular, for $\phi < 1$ we have shown that the product strictly increases after a swap, meaning that the LPs share of the pool also increases compared to the model where $\phi = 1$.

2. **Economic analysis of fee-bearing AMMs**
   We have established how trading fees affect key economic properties. In particular, in Section 3.3 we have found out that the fee-adjusted constant product swap rate is still ***output-bounded*** and ***strictly monotonic***, which ensures that the AMM's reserves are never drained upon a swap action and that the swap rate behaves predictably as reserves change.

   We also proved that it satisfies our own definition of ***additivity***, showing that swapping large aggregated amounts of tokens is more profitable than splitting it into smaller trades.

   In Section 3.4 we reasoned about the internal rate and its relationships with the external rate and swap rate, providing us with the necessary groundwork to find the solution to the arbitrage problem, and analyzing the differences from the one presented in [8], where it coincided with the equilibrium value.

Finally, in Section 3.5 we reasoned about the arbitrage problem and how to solve it, starting from the approach used in [8], finding the equilibrium value in Theorem 3.5.1. We also discovered that, in opposition to what was found in the foundational model, in this fee-bearing model, this value is not the optimal one, as proved in Lemma 3.5.4. Such optimal value is x_{max}, presented in Lemma 3.5.5. Finally, we also proved the uniqueness of both the equilibrium value and the optimal one.

3. **Lean 4 Formalization**
   Every definition, lemma and theorem that can be found in this work has been encoded in Lean 4, meaning that all the pen-and-paper proofs from this thesis are backed-up by rigorous machine-checked proofs. This not only ensures the mathematical correctness of this work, but also extends the already existing reusable library for future work on formalizing AMM properties.

4.1 Uniswap v2's comparison

In this section we analyze the differences and similarities between the swap used in this work's model, and the swap used in Uniswap v2's implementation. The Uniswap one is slightly more complicated than the one presented in this model, and the main differences are:

1. Uniswap allows to swap both tokens τ_0 and τ_1 at the same time, while our model allows only one atomic token type as input.

2. In Uniswap, when calling the function, the input is not the amount of tokens that the user wants to swap, but instead the maximum amount of tokens that the user expects to receive. The input token amount is then computed internally in the swap function.

Uniswap v2 is architected in two main modules, the *core* and the *periphery*. The core module only contains the minimal and critical contracts, while the periphery holds all the helper and router contracts that integrate and extend the core functionalities. It is worth mentioning that, according to Uniswap v2's guide on implementing a swap in a contract[1], it is recommended not to use the `swap` primitive from the core module, but instead to use the Router[2] from the periphery module. This is because the Router provides a set of methods that allow users to safely swap the desired tokens. For example, in our model, a swap action $\mathsf{A} : \mathsf{swap}(x, \tau_0, \tau_1)$ means that A wants to swap exactly an amount x of τ_0 tokens, and expects $y = x \cdot SX_\phi(x, r_0, r_1)$ of τ_1 tokens as output. For this scenario, the Router provides a function `swapExactTokensForTokens` which does exactly what we want. To help the reader understand the similarities and differences from our implementation to the one of Uniswap v2, suppose that we want to perform a swap $\mathsf{A} : \mathsf{swap}(x, \tau_0, \tau_1)$ in a state $\Gamma = \{r_0 : \tau_0, r_1 : \tau_1\} \mid \Delta$. Then, we explain how this swap is executed using Uniswap Router02's `swapExactTokensForTokens`, analyzing the variables and function calls being made, up until the swap primitive presented in Listing 4.5.

Listing 4.1 shows `swapExactTokensForTokens`'s implementation. In particular, we can see that it takes as input:

1. `amountIn`: this corresponds to our x in the swap

2. `amountOutMin`: this specifies the minimum amount of output tokens that the user A expects. In this work's model, we assume that this value is zero, as the rule [SWAP] does not express a minimum amount of tokens that the user can expect upon a swap.

[1] https://docs.uniswap.org/contracts/v2/guides/smart-contract-integration/trading-from-a-smart-contract
[2] https://github.com/Uniswap/v2-periphery/blob/master/contracts/UniswapV2Router02.sol

```
1  function swapExactTokensForTokens(
2          uint amountIn,
3          uint amountOutMin,
4          address[] calldata path,
5          address to,
6          uint deadline
7      ) external virtual override ensure(deadline) returns (uint[] memory amounts) {
8          amounts = UniswapV2Library.getAmountsOut(factory, amountIn, path);
9          require(amounts[amounts.length - 1] >= amountOutMin, 'UniswapV2Router:
               INSUFFICIENT_OUTPUT_AMOUNT');
10         TransferHelper.safeTransferFrom(
11             path[0], msg.sender, UniswapV2Library.pairFor(factory, path[0], path
                   [1]), amounts[0]
12         );
13         _swap(amounts, path, to);
14 }
```

Listing 4.1: Router02's swapExactTokensForTokens

3. `path`: this is the array of token addresses that the trade goes through. The first of these is the input token, and the last one is the output token. So, in our case this is going to be simply $[\tau_0, \tau_1]$.
4. `to`: this is the address that receives the final output tokens. In our case, this address coincides with A, the user performing the swap.
5. `deadline`: this is a timestamp after which the router reverts the swap if it has not been included in a block yet. This is mainly to prevent bad exchange rates if the transaction takes too long to be approved. Since the current model does not represent the transaction mempool, this feature is omitted.

To better understand what `swapExactTokensForTokens` does in its body, we first explain how the output amount y is computed, then break down how the transfer of the input tokens amount x happens, and finally analyze the swap primitive call.

4.1.1 Output amount

In line 8, the function calls the library function `getAmountsOut` [3], the code of which is reported in Listing 4.2. This function computes an array of amounts of output tokens depending on how many there are specified in the parameter `path`, where the first element of the array is the input token amount `amountIn`. In our case, since we are trading in the pool $\{r_0 : \tau_0, r_1 : \tau_1\}$ and we already explained that `path` $= [\tau_0, \tau_1]$, we will have `amounts[0]` $=$ `amountIn` and `amounts[1]` $=$ `getAmountOut(amounts[0], reserveIn, reserveOut)` where `reserveIn` and `reserveOut` are exactly our r_0 and r_1.

The function `getAmountOut` [4](Listing 4.3) is where the constant product swap rate is implemented. In comparison with our model, this returns exactly an amount $y = x \cdot SX_\phi(x, r_0, r_1)$ of τ_1 tokens. Analyzing the code line-by-line, the variables being computed are `amountInWithFee` $= x \cdot 997$, `numerator` $= x \cdot 997 \cdot r_1$, `denominator` $= r_0 \cdot 1000 + x \cdot 997$ and `amountOut` $= y$. Putting everything together:

$$y = \frac{\texttt{numerator}}{\texttt{denominator}} = \frac{x \cdot 997 \cdot r_1}{r_0 \cdot 1000 + x \cdot 997}$$
$$= \frac{1000 \cdot (x \cdot 0.997 \cdot r_1)}{1000 \cdot (r_0 + 0.997 \cdot x)} = \frac{x \cdot 0.997 \cdot r_1}{r_0 + 0.997 \cdot x}$$

[3] https://github.com/Uniswap/v2-periphery/blob/master/contracts/libraries/UniswapV2Library.sol#L62-L70
[4] https://github.com/Uniswap/v2-periphery/blob/master/contracts/libraries/UniswapV2Library.sol#L43-L50

```
function getAmountsOut(address factory, uint amountIn, address[] memory path)
    internal view returns (uint[] memory amounts) {
        require(path.length >= 2, 'UniswapV2Library: INVALID_PATH');
        amounts = new uint[](path.length);
        amounts[0] = amountIn;
        for (uint i; i < path.length - 1; i++) {
            (uint reserveIn, uint reserveOut) = getReserves(factory, path[i], path
                [i + 1]);
            amounts[i + 1] = getAmountOut(amounts[i], reserveIn, reserveOut);
        }
    }
```

Listing 4.2: getAmountsOut

```
function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut) internal
    pure returns (uint amountOut) {
        require(amountIn > 0, 'UniswapV2Library: INSUFFICIENT_INPUT_AMOUNT');
        require(reserveIn > 0 && reserveOut > 0, 'UniswapV2Library:
            INSUFFICIENT_LIQUIDITY');
        uint amountInWithFee = amountIn.mul(997);
        uint numerator = amountInWithFee.mul(reserveOut);
        uint denominator = reserveIn.mul(1000).add(amountInWithFee);
        amountOut = numerator / denominator;
    }
```

Listing 4.3: getAmountOut

which is exactly the constant product swap rate from Definition 2.4.1 with $\phi = 0.997$, the trading fee hard-coded in Uniswap v2.

Going back to line 8 of Listing 4.1, `amounts` will have the input amount x for the first element, and the output amount $y = x \cdot SX_\phi(x, r_0, r_1)$ that we just computed as the second one. In line 9, this safety check ensures that the output amount y is at least what the user A expects to receive as a minimum amount of output tokens. If not, the transaction is reverted.

4.1.2 Tokens transfer

In lines 10-11, we see the transfer of x τ_0 tokens into the contract of the pair $\{\tau_0, \tau_1\}$ before calling the function `_swap`. This is done because the `_swap` function expects the input tokens to be already transferred into the pair. The code of this function is in Listing 4.4. This is the function that prepares the parameters to call the `swap` primitive, illustrated in Listing 4.5. In particular, we can see that in line 6 the function instantiates `amount0Out` and `amount1Out`, which will be the parameters passed to the `swap` primitive. In this case, depending on the ordering of the tokens τ_0 and τ_1 in the AMM pair, one of `amount0Out` and `amount1Out` will be instantiated with our output amount y, and the other one to zero, since we do not expect two output amounts, but only one of τ_1 tokens. For simplicity, suppose from now on that in our case `amount0Out` $= 0$, since we are swapping τ_0 tokens.

Next, since we are dealing with only one token pair, when the function computes `to`, it is immediately set to `_to` (which in our case is A's wallet). The final line represents the call to the `swap` primitive, where

1. `amount0Out` is zero, as already explained, since we are swapping τ_0 tokens, and we don't expect to receive any τ_0 tokens as output.
2. `amount1Out` is equal to $y = SX_\phi(x, r_0, r_1)$.
3. `to` represents A's wallet.

```
function _swap(uint[] memory amounts, address[] memory path, address _to) internal
    virtual {
        for (uint i; i < path.length - 1; i++) {
            (address input, address output) = (path[i], path[i + 1]);
            (address token0,) = UniswapV2Library.sortTokens(input, output);
            uint amountOut = amounts[i + 1];
            (uint amount0Out, uint amount1Out) = input == token0 ? (uint(0),
                amountOut) : (amountOut, uint(0));
            address to = i < path.length - 2 ? UniswapV2Library.pairFor(factory,
                output, path[i + 2]) : _to;
            IUniswapV2Pair(UniswapV2Library.pairFor(factory, input, output)).swap(
                amount0Out, amount1Out, to, new bytes(0)
            );
        }
    }
```

Listing 4.4: Router02's swap

4.1.3 Swap primitive and Invariant check

Finally, we investigate the function `swap` in Listing 4.5. As previously stated, this function is slightly complicated, so we only want to give an intuition on its behavior for our example case. We begin by locating where our input and output amounts x and y are located/computed inside the function:

1. The input amount x is computed inside the function in line 18, called `amount0In`. Remember that inside the Router's `swapExactTokensForTokens` we have already performed a safe transfer of this input amount x. Hence, the function can simply compute `amount0In` as new reserves of τ_0 called `balance0` minus the old reserves, called `_reserve0`. Intuitively, since we did not transfer any τ_1 token to the pair, then `amount1In` is zero, as expected.

2. The output amount y is passed as the parameter `amount1Out` like previously said. Also, remember that the other parameter `amount0Out` is zero, from the computations made in the function `_swap`.

There is only one thing left to check: the invariant. So far, we have not found anything that checks that $(r_0 + x)(r_1 - y) \geq r_0 r_1$. This is done in an even stricter way inside the `swap` function. In line 22-23 the function declares `balance0Adjusted` and `balance1Adjusted`, which are computed as:

$$\texttt{balance0Adjusted} = (r_0 + x) \cdot 1000 - 3x$$

and

$$\texttt{balance1Adjusted} = (r_1 - y) \cdot 1000 - 3 \cdot 0 = (r_1 - y) \cdot 1000$$

Recall that at this point of the contract, `balance0` and `balance1` are respectively r_0 and r_1 after the transfers have occurred, i.e. $r_0 + x$ and $r_1 - y$. Then in line 24, the `require` specifies the invariant check:

$$\texttt{balance0Adjusted} \cdot \texttt{balance1Adjusted} \geq r_0 r_1 \cdot 1000^2$$

At first sight this check might seem very different from the one of our model. However, we can show that it is almost identical:

$$\begin{aligned}
&\texttt{balance0Adjusted} \cdot \texttt{balance1Adjusted} \geq r_0 r_1 \cdot 1000^2 \\
\iff &((r_0 + x) \cdot 1000 - 3x)((r_1 - y) \cdot 1000) \geq r_0 r_1 \cdot 1000^2 \\
\iff &(r_0 + x - 0.003x)(r_1 - y) \geq r_0 r_1
\end{aligned}$$

```
function swap(uint amount0Out, uint amount1Out, address to, bytes calldata data)
    external lock {
    require(amount0Out > 0 || amount1Out > 0, 'UniswapV2:
        INSUFFICIENT_OUTPUT_AMOUNT');
    (uint112 _reserve0, uint112 _reserve1,) = getReserves(); // gas savings
    require(amount0Out < _reserve0 && amount1Out < _reserve1, 'UniswapV2:
        INSUFFICIENT_LIQUIDITY');

    uint balance0;
    uint balance1;
    { // scope for _token{0,1}, avoids stack too deep errors
    address _token0 = token0;
    address _token1 = token1;
    require(to != _token0 && to != _token1, 'UniswapV2: INVALID_TO');
    if (amount0Out > 0) _safeTransfer(_token0, to, amount0Out); // optimistically
        transfer tokens
    if (amount1Out > 0) _safeTransfer(_token1, to, amount1Out); // optimistically
        transfer tokens
    if (data.length > 0) IUniswapV2Callee(to).uniswapV2Call(msg.sender, amount0Out
        , amount1Out, data);
    balance0 = IERC20(_token0).balanceOf(address(this));
    balance1 = IERC20(_token1).balanceOf(address(this));
    }
    uint amount0In = balance0 > _reserve0 - amount0Out ? balance0 - (_reserve0 -
        amount0Out) : 0;
    uint amount1In = balance1 > _reserve1 - amount1Out ? balance1 - (_reserve1 -
        amount1Out) : 0;
    require(amount0In > 0 || amount1In > 0, 'UniswapV2: INSUFFICIENT_INPUT_AMOUNT'
        );
    { // scope for reserve{0,1}Adjusted, avoids stack too deep errors
    uint balance0Adjusted = balance0.mul(1000).sub(amount0In.mul(3));
    uint balance1Adjusted = balance1.mul(1000).sub(amount1In.mul(3));
    require(balance0Adjusted.mul(balance1Adjusted) >= uint(_reserve0).mul(
        _reserve1).mul(1000**2), 'UniswapV2: K');
    }

    _update(balance0, balance1, _reserve0, _reserve1);
    emit Swap(msg.sender, amount0In, amount1In, amount0Out, amount1Out, to);
}
```

Listing 4.5: Uniswap v2's swap

$$
\begin{aligned}
&\iff (r_0 + 0.997x)(r_1 - y) \geq r_0 r_1 \\
&\iff (r_0 + \phi x)(r_1 - y) \geq r_0 r_1 && \text{with } \phi = 0.997
\end{aligned}
$$

This check is even more strict than the one in Eq. 2.1, as this only takes into account the fee-reduced input ϕx. This check clearly implies the one in Eq. 2.1, as $r_0 + \phi x < r_0 + x$.

4.2 Discussion and Future Work

Although we were able to prove all of the lemmas and theorems above, the current Lean 4 library that implements this model and all the relative proofs has some limitations.

The type used to represent positive amounts in the model, i.e. some $x \in \mathbb{R}_{>0}$, poses some significant challenges. This type is defined as a ***subtype*** of the already existing — and well supported — Reals $\mathbb{R}$. This approach has some positive sides, such as guaranteeing that the results of some lemmas are strictly positive (for example, it makes sure that for an initialized AMM the reserves are always strictly positive). This modelling choice also comes with a few negative aspects. For example, positive reals by definition do not have a zero element, hence they cannot form a ring, since it lacks additive identity and inverse. This might seem like a minor issue, but it greatly impacts some of the Lean 4 tactics that are usually used to perform algebraic manipulation (e.g. `ring-nf`). This means that automation in proofs is limited and not always useful.

Another issue that arose during the development of the proofs presented in this work is related to the simplification lemmas. There are a number of already existing simplification lemmas that are not always meant to be used by Lean 4's simplifier. This also greatly impacts automation, as sometimes the prover is stuck due to infinite loops caused by misleading simplification lemmas. This is something that is worth investigating, as it could improve automation significantly.

Finally, we mentioned in the proof of Theorem 3.5.5 that a natural way of proving this Theorem is to follow the function analysis path. However, to the best of our knowledge, this is not possible due to the current implementation of Mathlib 4's function analysis tools and our Lean 4 model implementation. To find the maximum value of the gain function, we would have used the Analysis module from Mathlib 4. In particular, we would have used the fact that the function is concave, and so a local maximum of a concave function is also a global maximum. This is formalized by the theorem `of_isLocalMaxOn_of_concaveOn`[5]. However, since the gain function's domain is $\mathbb{R}_{>0}$, we cannot express this condition, as the theorem specifically requires a function domain that is an additive commutative monoid. There might be other ways of solving this or workarounds, but due to time constraints we did not explore this proof outline further. .

We envision the following directions for future research:

1. **Guarded transactions and Slippage Protection**: A first extension that would bring this model closer to the actual implementation of Uniswap v2 can be formalizing the parameters of the function `swapExactTokensForTokens` (Section 4.1) that the current model is not able to express, i.e. `amountOutMin` and `deadline`. The first one would ensure a minimum amount of tokens to the user that performs the swap, while the second one is mainly to prevent the transaction to stay pending for a long period, reducing the risk of possible slippage.

2. **Concentrated Liquidity and Multi-Fee tiers**: Uniswap v3 [2] introduces so-called *concentrated liquidity*: instead of depositing liquidity across the whole range of the internal exchange rate, in Uniswap v3 LPs can specify an internal exchange rate range in which their liquidity earn fees. This allows for different strategies for LPs which can increase their gain but also the risk. Another feature that was introduced in Uniswap v3 is the one of multiple fee tiers. Uniswap v2 has a flat fee of 0.3% on every pool, while in Uniswap v3 it is possible to decide on four discrete tiers upon the creation of the pool, which are 0.01%, 0.05%, 0.3% and 1%. Clearly, a higher fee tier benefits the LPs, but could also result in a less attractive pool to trade on for arbitrageurs. Formalizing these features in the current model would bring it closer to the state-of-art of AMM's implementation.

3. **Liquidity Provider Analysis**: This work mainly focuses on the arbitrageur's point of view, i.e. what are their incentives and what are the best possible values that they can swap to maximize their gain. However, we have not analyzed how the trading fees impact the LPs: we know that their share in the pool's reserves increases compared to the model without the trading fee, however it could still be interesting to measure this improvement and analyzing the impact of trading fees from the liquidity provider's side as well.

4. **Lean 4 Library Enhanced Automation**: This work's implementation can be greatly improved by enhancing the proof's automation. As previously explained, the automa-

[5] https://leanprover-community.github.io/mathlib4_docs/Mathlib/Analysis/Convex/Extrema.html#IsMaxOn.of_isLocalMaxOn_of_concaveOn

tion is greatly affected by a number of reasons, so an improvent could be trying to modify — or completely change — the subtype of positive real numbers which is vastly used in the library, to review the simplification lemmas and so on.

Bibliography

[1] Hayden Adams, Noah Zinsmeister, and Dan Robinson. *Uniswap v2 Core*. https://app.uniswap.org/whitepaper.pdf. Accessed: May 12, 2025. Mar. 2020.

[2] Hayden Adams et al. *Uniswap v3 Core*. https://app.uniswap.org/whitepaper-v3.pdf. Accessed: May 12, 2025. Mar. 2021.

[3] *SushiSwap token pair implementation*. https://github.com/sushiswap/sushiswap/blob/94ea7712daaa13155dfab9786aacf69e24390147/contracts/uniswapv2/UniswapV2Pair.sol. 2021.

[4] Fatemeh (Nazanin) Mottaghi and Bertram I. Steininger. *Total Value Locked Volatility in DeFi: Empirical Evidence from Market Crashes*. https://ssrn.com/abstract=5161601. Accessed: May 12, 2025. 2025.

[5] Daniele Pusceddu and Massimo Bartoletti. "Formalizing Automated Market Makers in the Lean 4 Theorem Prover". In: *5th International Workshop on Formal Methods for Blockchains, FMBC 2024, April 7, 2024, Luxembourg City, Luxembourg*. Ed. by Bruno Bernardo and Diego Marmsoler. Vol. 118. OASIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 5:1–5:13. DOI: 10.4230/OASICS.FMBC.2024.5. URL: https://doi.org/10.4230/OASIcs.FMBC.2024.5.

[6] Leonardo de Moura and Sebastian Ullrich. "The Lean 4 Theorem Prover and Programming Language". In: *Automated Deduction – CADE 28*. Vol. 12699. Lecture Notes in Computer Science. Springer, Cham, 2021, pp. 625–635. DOI: 10.1007/978-3-030-79876-5_37. URL: https://lean-lang.org/papers/lean4.pdf.

[7] Oliver Emil Bøving. *cite-lean*. https://github.com/oeb25/cite-lean. Accessed: 2025-05-12. 2025.

[8] Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch-Lafuente. "A theory of Automated Market Makers in DeFi". In: *Log. Methods Comput. Sci.* 18.4 (2022). DOI: 10.46298/LMCS-18(4:12)2022. URL: https://doi.org/10.46298/lmcs-18(4:12)2022.

[9] Jason Milionis, Ciamac C. Moallemi, and Tim Roughgarden. "Automated Market Making and Arbitrage Profits in the Presence of Fees". In: *Financial Cryptography and Data Security, FC 2024, Lecture Notes in Computer Science, vol. 14744*. Ed. by Jeremy Clark and Elaine Shi. Springer, Cham, 2025, pp. 159–171. DOI: 10.1007/978-3-031-78676-1_9.

[10] Guillermo Angeris and Tarun Chitra. "Improved Price Oracles: Constant Function Market Makers". In: *ACM Conference on Advances in Financial Technologies (AFT)*. https://arxiv.org/abs/2003.10001. ACM, 2020, pp. 80–91. DOI: 10.1145/3419614.3423251.

[11] Jason Milionis et al. "Automated Market Making and Loss-Versus-Rebalancing". In: *arXiv preprint arXiv:2208.06046* (2022). DOI: 10.48550/arXiv.2208.06046. URL: https://arxiv.org/abs/2208.06046.

[12] Robin Fritsch, Samuel Käser, and Roger Wattenhofer. "The Economics of Automated Market Makers". In: *Proceedings of the 4th ACM Conference on Advances in Financial*

Technologies (AFT '22). Cambridge, MA, USA: ACM, 2022, pp. 102–110. ISBN: 978-1-4503-9861-9. DOI: 10.1145/3558535.3559790.

[13] *Uniswap token pair implementation*. https://github.com/Uniswap/uniswap-v2-core/blob/4dd59067c76dea4a0e8e4bfdda41877a6b16dedc/contracts/UniswapV2Pair.sol. 2021.

[14] *Mooniswap implementation*. https://github.com/1inch-exchange/mooniswap/blob/02dccfab2ddbb8a409400288cb13441763370350/contracts/Mooniswap.sol. 2020.

[15] Wikipedia contributors. *Concave function*. https://en.wikipedia.org/wiki/Concave_function. [Online; accessed 11-May-2025]. n.d.

Technical
University of
Denmark

Richard Petersens Plads, Building 324
2800 Kgs. Lyngby
Tlf. 4525 1700

www.compute.dtu.dk