The goal of this post is to introduce the concept of Sequencing Rules and provide a possible sketch of what a Sequencing Rule-enabled EVM would look like. Sequencing Rules offer a rational solution to both Ethereum's MEV and protocol innovation mindshare loss problems.

Sequencing Rules: The Why

Ethereum is facing a MEV crisis. FUD is building on Twitter as block builders grow in their influence over the chain's functioning. Block builders are collecting an uncomfortable amount of capital with each block, yet they provide little to negative value in return. This behavior makes sense: value has been left on the table, so why shouldn't they collect it? A rational way to stop the growing block builder oligopoly is to enable Ethereum's on-chain applications to fully capture the value that they create.

So far, discussions on fixing block builders' power have focused only on lower-level structural L1 changes1,2,3

, leaving out the actual source of value creation and leakage: the on-chain applications themselves. The best entity to decide how to fairly distribute the value created from ordering is the entity that is producing the value. For the sake of Ethereum's long-term success, on-chain applications need to be able to express and control how their transactions are sequenced with supporting L1/L2 infrastructure.

The rest of this post will:

- Introduce the concept of Sequencing Rules
- Provide a sketch of what native EVM sequencing support could look like
- Provide a proof-of-concept example of an AMM that auctions off the right of the first swap on Ethereum

Sequencing Rules: The What

Sequencing rules are not a new concept but have recently seen increased usage by blockchain developers.

Sequencing rules can be conceptualized as follows:

1. On-chain applications decide how they want their transactions to be sequenced.
2. Blocks are built that respect that sequencing.

Blockchain applications can benefit from this programming pattern due to blockchains' inherent discrete view of time. During the 12 seconds between Ethereum's blocks, the outside world's valuation of different pieces of state is in flux, creating arbitrage value in the ability to act first on those pieces of state. Block builders make money by exploiting this in the form of acts like CEX-DEX arbitrage. This phenomenon is also observed in traditional finance (TradFi) with the high-speed arms race between high-frequency trading (HFT) firms, which operate in a similar scenario but on a millisecond timeframe instead of seconds.

As the field of protocol development matures, protocols are starting to manage these value-pressurized pieces of state with varying degrees of success. Examples vary but all showcase the benefits of managing one's own sequencing:

- Penumbra, a new L1 DEX, enforces ordered batch execution of all business logic, enabling all users to benefit equally from the accrued time arbitrage value.
- HyperLiquid's new L1 prioritizes retail and makers over toxic takers by running cancels and post-only orders first.
- Skip's Slinky project surfaces oracle data at the block level to ensure all applications can utilize fresh oracle data.
- Oval auctions off the right to be the first person to act on updated oracles information.
- Years of research1,2,3

have been conducted on how to implement first-swap auctions for AMMs on Ethereum, though no production-grade result is running yet.

Sequencing rules for blockchain applications are natural, yet Ethereum does not offer native solutions to enable developers to program their applications accordingly.

Sequencing Rules: The How

As seen in the list above, protocols are starting to migrate to other ecosystems with novel L1 implementations because their sequencing needs are not being met on Ethereum. Instead of Ethereum's on-chain protocols capturing the time-accrued value, block builders are. Ethereum's EVM needs sequencing rules.

Ethereum's EVM programming environment is different from an L1 that is specialized for a specific application because it is a shared general execution environment. Applications and transactions share both block space and state with each other. As a result, anyone implementing sequencing rules for an EVM should consider:

- On-chain applications cannot assume a piece of state's value will not change unless it is explicitly controlled.
- Not all on-chain applications need sequencing.
- Not all actions within an application need sequencing.

I will now lay out a rough draft version of how one could implement sequencing rules for Ethereum. In order to meet the above considerations, I propose adding two new transaction types:

1. A new 'bundle' transaction type. This transaction would be signed by an EOA and would contain a payload of ordered transactions. We'll call this EOA a bundler. This 'transaction' itself does not trigger on-chain execution, it simply just orders other transactions.
2. A new transaction type that specifies the bundler (just an EOA) that is able to include the transaction in a bundle.

[

IMG_0025

3239×2350 261 KB

](https://ethresear.ch/uploads/default/original/3X/b/3/b376ab484f0a2eaa09bcb254d7f12439f442621d.png)

EVM block construction would be modified to enforce the following rules:

- A bundler can only have a single bundle transaction included in a block, and optionally, a bundle is only valid in a certain block. Some business applications would benefit from being able to run bundles in only certain blocks.
- A bundle can specify a bundler that is able to include it in a bundle, for meta-bundle programming.
- A transaction can only be included in a bundle if it specifies bundler, and if a transaction specifies a bundler, the transaction can only be included in a block if it is in a bundle. This is to prevent bundles from clashing with each other, which would cause builders to have extra incentive to include one bundler over another.
- Unlike Ethereum's current block builder's bundle interface, transactions in a bundle do not have any non-revert guarantees. On-chain application logic must handle possible reverts. This is again to prevent bundles from clashing with each other.

The EVM itself would be modified to include the new following global variables:

- bundle.signer

: the EOA who signed the bundle. Useful for on-chain applications to designate sequencing rights to a specific off-chain entity. The zero address if not in a bundle.

- bundle.number

: the global number of the bundle's ordering. Useful for meta-bundle programming. Zero if not in a bundle.

- bundle.tx_number

: which number this transaction is in the bundle. Useful for on-chain applications to enable special logic depending on transaction sequence. Zero if not in a bundle.

These new EVM rules and global variables could be a strong starting point for on-chain applications to express how they'd like to be sequenced.

There are numerous security and caveats to consider in this implementation. EVM bundle creators would need to take care to only sequence on state that they directly control from the start of the block. This is because all other state is potentially manipulatable by outside entities. Bundle creators would have to assume that all transactions could revert due to insufficient gas funds, that any piece of unprotected state could be changed, and that any starting point besides its own external functions are non-deterministic. Sequencing rule assumptions about mutable state could be a new class of audit vulnerabilities.

Additionally, these rules or language expansions do not preclude block builders existing or making money. Bundlers could construct multiple bundles and rely on block building services to take advantage of the highest result. Inter-bundle arbitrage could be a new solver target. Analogous to the EVM global variables proposed above, tx.number

could be included to allow block builders to explicitly sell the right to be the first transaction included.

Notably, this implementation lacks specification on how on-chain applications would construct bundles. This would be up to the applications to chose between different trust and privacy tradeoffs. Some options would include using:

- Flashbot's Suave TEE enabled decentralized platform for business applications needing confidential compute.
- A sequencing layer, like Astria, to construct generic bundles.

Sequencing Rules: An Example

I came up with the above Ethereum compatible sequencing rule proposal while trying to program a first swap auction for a AMM on Ethereum. This proof of concept aims to implement the sequencing rule 'the first swap should be auctioned' for the sake of rerouting some of the CEX-DEX arbitrage back to the pool's LPs. It utilizes block builder's current bundle infrastructure to craft bundles and inserts logic on-chain via a signed transaction to enforce that the bundle's sequence is respected. This proof of concept uses Flashbot's SUAVE TEE enabled programming environment to craft bundles for the AMM in a trust-reduced manner. This implementation could be easily rewritten with the above EVM native bundle proposal with much better execution costs and guarantees.

I wrote this post because I'm a frustrated protocol developer who wants to capture the maximal value for my actual users. These are just my thoughts on how we can potentially make "Ethereum Great Again"

.

Please leave comments if you have them!

Thank you to my teammates Itamar and Vishesh at Astria for agreeing to talk about auctions for way too long, The Latest in Defi Research for providing the community to iterate on these thoughts, and to the Suave team for letting me learn how to build bundles on their platform.

*Note: these @'s have no endorsement or involvement in this post, these words and opinions are my own :).