Arbitrum for Stacks

Or, how I stopped worrying and learned to love Clarity type casting!

But why?

- Blockchains are great! But they have bandwidth limits. What if we had many blockchains and they could interact with each other and share trust? *Layer-2*
- But trust between layers isn't automatic! Miners of Stacks aren't punished if miners of a layer-2 misbehave.
- What if users could punish layer-2 miners when they misbehave*? optimistic rollups.*
- Arbitrum is an optimistic rollups protocol, which already has an EVM compiler
- Arbitrum defines a virtual machine semantics for running code – if we can implement this VM in Clarity, then the Stacks chain can verify fraud proofs for Arbitrum layer 2s.

How optimistic rollups work

- Layer 2 miners produce a block, and sign a hash of that blocks “state”, broadcasting it to a layer 1 contract.
- If they misbehave in Block N+1, users can show how the state of Block N, and the transactions in Block N+1 should lead to state *s*, but the miner signed *s’*
- A layer 1 contract needs to be able to confirm *s* is expected
  - Do this by actually “running” the layer 2 code in a layer 1 contract
- Arbitrum defines a virtual machine semantics for running code – if we can implement this VM in Clarity, then the Stacks chain can verify fraud proofs for Arbitrum layer 2s.

This project: Clarity AVM

- Implement a proof-of-concept virtual machine for Arbitrum in a Clarity smart contract.
  - Took some shortcuts:
    - I didn’t implement all the instructions (not enough time? boredom? who could say.)
    - Didn’t implement arbitrum’s wire formats for hashing, serialization. Just used Clarity’s.
- Test with a simple program or two.
- Needed Clarity2, so also hacked at clarinet until things worked
  - Side note: discovered 2 bugs in Clarity2! PR: https://github.com/stacks-network/stacks-blockchain/pull/3226

Next steps

- Testing with EVM
  - Arbitrum has an EVM to AVM compiler – integrating EVM applications with this will require linking that compiler to the input interface of the Clarity contract
- Integration with Layer 1
  - All this implements is the AVM – optimistic rollups need to use this information for fraud proofs, slashing. Those will L1 contract integration to deserialize fraud proofs, implement punishment
- Performance
  - Contract uses MARF for almost every operation. Could the data stacks instead be local variables?
- Correctness
  - This is a big pile of nearly entirely untested Clarity code.