One Step Proof Assumptions

The One Step Proof (OSP) implementation makes certain assumptions about the cases that can arise in a correct execution. This documents those assumptions about what's being executed.

If a case is "unreachable", that is, the case is assumed to never arise in correct execution, then the OSP can implement any instruction semantics in that case.

- In a challenge between malicious parties, any case can arise. The challenge protocol must do
- something safe in every case. But the instruction semantics can be weird in such cases because
- if both parties to a challenge are malicious, the protocol doesn't care who wins the challenge.
- In a challenge with one honest party, the honest party will never need to one-step prove an
- unreachable case. The honest party will only assert correct executions, so it will only have to
- prove reachable cases.
- In a challenge with one honest party, the dishonest party could assert an execution that transitions
- into an unreachable case, but such an execution must include an invalid execution of a reachable case
- earlier in the assertion. Because a challenge involving an honest party will eventually require an OSP
- over the first instruction where the parties disagree, the eventual OSP will be over the earlier point
- of divergence, and not over the later execution from an unreachable case.

In general, some unreachable cases will be detectable by the OSP checker and some will not. For safety, the detectable unreachable cases should be defined by transition the machine into an error state, allowing governance to eventually push an upgrade to recover from the error. An undetectable unreachable case, if such a case were reached in correct execution, could lead to a security failure.

The following assumptions, together, must prevent an unreachable case from arising in correct execution.

The WAVM code is generated by Arbitrator from valid WASM

WAVM is the name of the custom instruction set similar to WASM used for proving. Arbitrator transpiles WASM code into WAVM. It also invokes wasm-validate fromwabt (the WebAssembly Binary Toolkit) to ensure the input WASM is valid. WAVM produced otherwise may not be executable, as it may try to close a non-existent block, mismatch types, or do any other number of invalid things which are prevented by WASM validation.

WAVM code generated from by Arbitrator from valid WASM is assumed to never encounter an unreachable case.

Inbox messages must not be too large

The current method of inbox hashing requires the full inbox message be available for proving. That message must not be too large as to prevent it from being supplied for proving, which is enforced by the inboxes.

The current length limit is 117,964 bytes, which is 90% of themax transaction size Geth will accept, leaving 13,108 bytes for other proving data.

Requested preimages must be known and not too large

WAVM has an opcode which resolves the preimage of a Keccak-256 hash. This can only be executed if the preimage is already known to all nodes, and can only be proven if the preimage isn't too long. Violations of this assumption are undetectable by the OSP checker.

The current length limit is 117,964 bytes for the reasons mentioned above. Here's a list of which preimages may be requested by Nitro, and why they're known to all parties, and not too large:

Block headers

Nitro may request up to the last 256 L2 block headers. The last block header is required to determine the current state, and blocks before it are required to implement theBLOCKHASH evm instruction.

This is safe as previous block headers are a fixed size, and are known to all nodes.

State trie access

To resolve state, Nitro traverses the state trie by resolving preimages.

This is safe as validators retain archive state of unconfirmed blocks, each trie branch is of a fixed size, and the only variable sized entry in the trie is contract code, which is limited by EIP-170 to about 24KB. Edit this page Last updatedonMar 19, 2024 Previous ChallengeManager Next WASM to WAVM