ETHOnline 2020 Hackathon

Composable

October 2020

1 Introduction

We define encoding

$$\hat{E} := \text{functionType} + \text{dataSize} + \text{callvalue} + \text{data}$$

where addition denotes concatenation, ie, non-standard packed encoding and

$$\begin{aligned} &\text{length}(\text{functionType} + \text{dataSize}) := 32 \\ &\rightarrow \text{dataSize} \in \text{Uint64} \end{aligned}$$

since functionType is well defined as a bytes24 value

$$\begin{aligned} &\text{functionType} := \text{contractAddress} + \text{functionSelector} \\ &\rightarrow \text{length}(\text{functionType}) = 24 \text{ bytes} \end{aligned}$$

and datasize specifies the length of data in individual bytes. The range of callvalue is the range of Ether values, so it is encoded in place as a Uint256. We make no assumptions about the lastly appended data, other than knowing its size in individual bytes. Naively, I want to say that the encoding for data (input by user) shouldn't be modified by our encoding/decoding, but it is possible of course when dealing with low-level memory management. We plan to use Yul to further optimize the encoding and decoding functions.

1.1 Reasoning and Application

Consider an arbitrary function F having s length in bytes, being called between contracts via the **call** opcode, such that F is a message call within the scope of a single block, originating possibly from some EOA initiated transaction T. Given $\hat{E}$, it should be possible to pass an arbitrary amount of calls through a decoding contract by utilizing the **calldatasize** opcode. Since we can retrieve the size of the entire calldata at the onset of the message calls, and we know the dynamic arrays' lengths from decoding the first 32 bytes of each individual call, then we can calculate the end of the data since the offset to the arbitrary data in every case is always 64, which is the length of $\hat{E}$ sans the data bytes and callvalue. In effect, although we may have multiple dynamic encoded parameters, the fact that they are in sequence ensures that at any given time we only ever have one dynamic encoded type, therefore in every case of contract to contract message calls we have a deterministic encoding for arbitrary dynamic data.