Folidity - Safe Functional Smart Contract Language

Project Brief

Candidate
German Nikolishin
University of Southampton
gn2g21@soton.ac.uk

Supervisor
Prof. Vladimiro Sassone
University of Southampton
vsassone@soton.ac.uk

October 2023

Problem statement

With the rise of blockchain technologies, smart contracts (SC) allowed developers to create complex and resilient applications providing services to end users. However, there have been numerous instances of attacks associated with decentralized applications, involving re-entrance attacks, forced value sends, variable overflows, and incorrectly coded state checks.

This is a result of the dominating nature of procedural style in SC languages such as Solidity, Vyper, PyTeal, and others that make the state transition implicit and hidden from the developers. SC developers then need to opt in for formal verification tools such as KEVM or KAVM that prolong the development process and require specialised knowledge of formal verification.

Proposed solution

This project proposes the development of a functional SC language with an explicit state transition and model checks. SCs run in a restricted and sandboxed environment and take part in the state transition of the blockchain state machine. This provides a pure functional context that is suitable for a functional programming language.

Folidity intends to be an SC language with a pure functional programming style that allows developers to reason about their code as a combination of state transition functions. The language also enforces developers to describe storage and state transition functions using constraints and invariants that the compiler will use to formally verify the functional correctness and consistency of a storage model, inspired by Event-B model verification.

Folidity targets the Algorand Virtual Machine (AVM). AVM is famous for its stack-based assembly language, Teal, which provides fine control over execution. Programs on AVM are also reasoned in the form of stateful and stateless applications. This philosophy perfectly aligns with the programming model of Folidity. The most important benefit of opting for AVM is the fixed SC execution costs which is highly important for a prototype language.

Scope

The project involves syntax design, the development of a compiler that produces intermediate representation code in Teal, and technical analysis of a sample SC demonstrating and proving its functional safety.

Due to limited time, this project will only focus on a single-contract execution, leaving the support of cross-contract calls beyond the scope of the project.