Modeling EVM in the K framework

Deepak Kumar Everett Hildenbrandt Manasvi Saxena Zane Ma
University of Illinois Urbana-Champaign

ABSTRACT

Ethereum smart contracts are everywhere. In this work, we model the EVM in K, a language built specifically for extensible, formal verification of programming languages.

1. INTRODUCTION

Ethereum smart contracts are important, and verifying them is a valuable thing to know how to do.

2. BACKGROUND

The growing popularity of smart contracts has led to increased scrutiny of the security of smart contracts. Because contracts inherently involve money, bugs in smart contract programs can be devastating to parties involved. A example of such a catastrophe is the recent DAO attack [2], where $150 million of ether was stolen, prompting an unprecedented hard fork of the Ethereum blockchain. Many classes of bugs exist in Ethereum smart contracts, from transaction-ordering dependence to mishandled exceptions and even a contract running out of gas before full completion of a function.

Smart contracts are an attractive method for formal program verification, which theoretically can provide guarantees on smart contract execution safety, liveness, and other forms of application logic. In particular, K is an extensible semantic framework that is used to model programming languages, type systems, and formal analysis tools [4]. There a number of semantic frameworks to choose from - we are choosing to use K in our formal analysis primarily because of its simplicity and extensibility. For instance, adding additional pieces of state is typically as easy as adding in new entries into a configuration and updating the rules to match. We are also working closely with the developers of K (Manasvi and his lab mantain K) to get more nuanced feedback about the rigor and correctness of our semantic model.

3. METHODOLOGY

We used K and modeled the EVM, to say interesting things about EVM programs.

4. EVALUATION

We evaluated our model for correctness based on our written tests and by use of some contracts written for specific purposes on the Ethereum blockchain.

5. RELATED WORK

Luu et al. [3] formalized a subset of EVM semantics, and used a symbolic execution tool based on their formalized semantics to detect problems in 8519 existing smart contracts. Their semantics however, are incomplete, and can only be used to detect a limited class of problems. For instance, their semantics do not consider the role of gas in Ethereum transactions, and cannot be used to prove properties concerning gas on smart contracts.

There exists extensive literature on the use of language semantics for program verification. Stefanescu et al. [1] in their work presented a language independent verification framework for program verification. The verification infrastructure presented in their work however, has to be initialized with the operational semantics of the language. Hence, having complete semantics of EVM in K would allow for using existing infrastructure for smart contracts verification.

6. REFERENCES

[1] A. Ştefănescu, D. Park, S. Yuwen, Y. Li, and G. Roşu. Semantics-based program verifiers for all languages. In *Proceedings of the 31th Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'16)*. ACM, Nov 2016.

[2] P. Daian. Dao attack, 2016. http://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/.

[3] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making smart contracts smarter. Cryptology ePrint Archive, Report 2016/633, 2016. http://eprint.iacr.org/2016/633.

[4] G. Roşu and T. F. Şerbănuţă. An overview of the K semantic framework. *Journal of Logic and Algebraic Programming*, 79(6):397–434, 2010.