Formal investigation of the expressiveness of the Extended UTxO model

Laying the foundations for the formal verification of smart contracts

ORESTIS MELKONIAN, Utrecht University, The Netherlands

This report serves as the proposal of my MSc thesis, supervised by Wouter Swierstra from Utrecht University and Manuel Chakravarty from IOHK.

1 INTRODUCTION

Although blockchain technology has opened a whole array of interesting new applications (e.g. secure multi-party computation[Andrychowicz et al. 2014], fair protocol design fair[Bentov and Kumaresan 2014], zero-knowledge proof systems[Goldreich et al. 1991]), reasoning about the behaviour of such systems is an exceptionally hard task. This is partly due to their concurrent nature, but also the fiscal nature of the majority of the applications, which require a much higher degree of rigorousness compared to conventional IT applications.

The advent of smart contracts (programs that run on the blockchain itself) gave rise to another source of vulnerabilities. Since these (possibly Turing-complete) programs often deal with transactions of significant funds, it is of utmost importance that one can reason and ideally provide formal proofs about their behaviour in a concurrent/distributed setting.

Research Question. The aim of this thesis is to provide a mechanized formal model of an abstract distributed ledger equipped with smart contracts, in which one can begin to formally investigate the expressiveness of the extended UTxo model. Moreover, we hope to lay down firm grounds, onto which one can further conduct a formal comparison with account-based models used in Ethereum. Put concisely, the research question posed is:

> *How much expressiveness do we gain by extending the UTxO model?*
> *Is it as expressive as the account-based model used in Ethereum?*

Overview. Section 2 reviews some basic definitions related to blockchain technology and introduces important literature, which will be the main subject of study throughout the development of our reasoning framework. Section 3 describes the technology we will use to formally reason about the problem at hand and some key design decisions we set upfront. Section 4 presents the progress made thus far in terms of (mechanized) formal verification, as well as problems we have encountered and also expect along the way. Section 5 discusses next steps for the remainder of the thesis, as well as a rough estimate on when these milestones will be completed.

2 BACKGROUND

2.1 Distributed Ledger Technology: Blockchain

2.2 Smart Contracts

2.3 UTxO-based: Bitcoin

SCRIPT. ...

The BitML Calculus. Bad documentation ... scarcity of formal models ... bitML ...

Extended UTxO. ...

Author's address: Orestis Melkonian, Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, melkon.or@gmail.com.

2.4 Account-based: Ethereum

Solidity.

3 METHODOLOGY

3.1 Scope

... no cryptographic/implementation details ... ... focus on the big picture ...

3.2 Proof Mechanization

Through mechanization ... ... vs informal mathematics ...

It is exactly this side effect, that will allow us to discover edge cases and increase the confidence of the model under investigation.

As our proof development vehicle, we choose Agda...

3.3 Agda

... stay on a highly abstract level ... postulate cryptographic operations etc ...

Limitation. ... proof automation ala Coq ...

3.4 The IOHK approach

... rigorous methodology ... ... industry co-existing with academia ... ... formal verification (Agda/Coq) -> prototype/reference implementation (Haskell) -> production codebase (Haskell) ...

3.5 Functional first

... UTxO vs Account ... ... functional vs imperative ... ... dataflow vs ?? ...

4 PRELIMINARY RESULTS

4.1 Formal Model I: Extended UTxO

4.1.1 *Inherently-typed validity of transactions.*

4.1.2 *Scripts via Denotational Semantics.*

4.1.3 *Address space as module parameter.*

4.1.4 *Weakening Lemma.*

4.1.5 *Example.*

4.2 Formal Model II: BitML Calculus

4.2.1 *Contracts in BitML.*

4.2.2 *Small-step Semantics.* ... mention paper bug in [C-Control] ...

4.2.3 *Configurations modulo permutation.*

4.2.4 *Example.*

4.3 Expected Problems

... up to permutation -> quotient types -> homotopy type theory ...

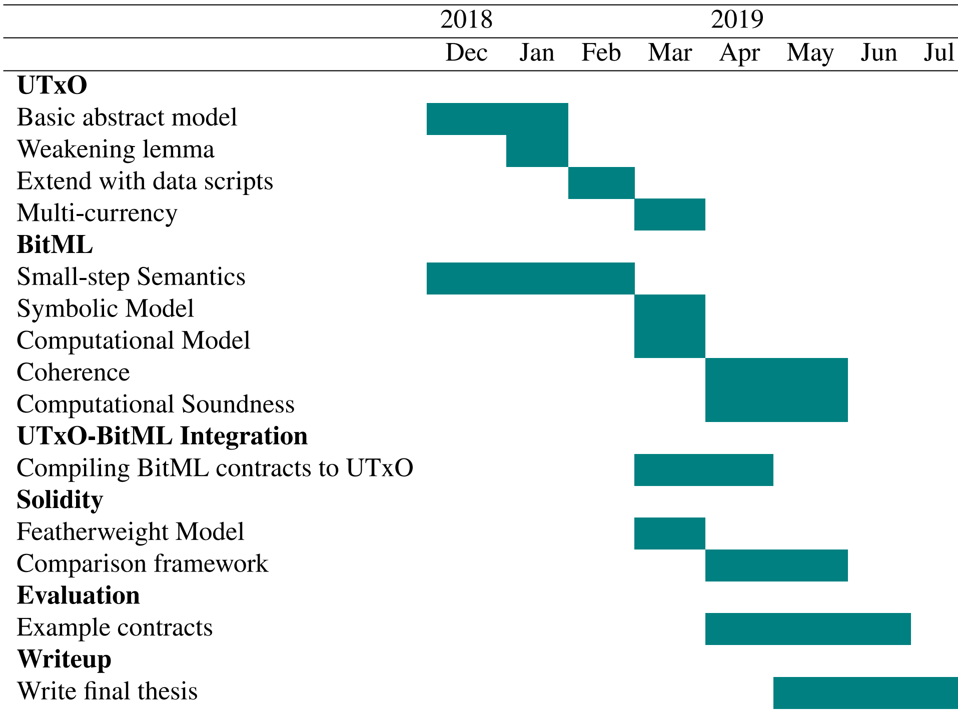

Fig. 1. My workplan.

5 PLANNING

5.1 Extended UTxO

... multi-currency ...

5.2 BitML Calculus

... symbolic runs ... computational runs ... coherence

5.3 UTxO-BitML Integration

5.4 Featherweight Solidity

5.5 Formal Comparison

... lots of examples ...

5.6 Proof Automation

5.7 Timetable

REFERENCES

2010. Script - Bitcoin Wiki. Retrieved 2/2019 from https://en.bitcoin.it/wiki/Script

2018. Formal verification of a Cardano wallet. Retrieved 2/2019 from https://cardanodocs.com/files/formal-specification-of-the-cardano-wallet.pdf

2019. The Extended UTxO Model. Retrieved 2/2019 from https://github.com/input-output-hk/plutus/blob/master/docs/extended-utxo/README.md

Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. 2014. Secure multiparty computations on bitcoin. In *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 443–458.

Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Jean-Christophe Filliatre, Eduardo Gimenez, Hugo Herbelin, Gerard Huet, Cesar Munoz, Chetan Murthy, et al. 1997. *The Coq proof assistant reference manual: Version 6.1*. Ph.D. Dissertation. Inria.

Massimo Bartoletti and Roberto Zunino. 2018. *BitML: a calculus for Bitcoin smart contracts*. Technical Report. Cryptology ePrint Archive, Report 2018/122.

Iddo Bentov and Ranjit Kumaresan. 2014. How to use bitcoin to design fair protocols. In *International Cryptology Conference*. Springer, 421–439.

Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cedric Fournet, A Gollamudi, G Gonthier, N Kobeissi, A Rastogi, T Sibut-Pinote, N Swamy, and S Zanella-Béguelin. 2016. Short paper: Formal verification of smart contracts. In *Proceedings of the 11th ACM Workshop on Programming Languages and Analysis for Security (PLAS), in conjunction with ACM CCS*. 91–96.

Vitalik Buterin et al. 2014. A next-generation smart contract and decentralized application platform. *white paper* (2014).

Oded Goldreich, Silvio Micali, and Avi Wigderson. 1991. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM (JACM)* 38, 3 (1991), 690–728.

Atsushi Igarashi, Benjamin C Pierce, and Philip Wadler. 2001. Featherweight Java: a minimal core calculus for Java and GJ. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 23, 3 (2001), 396–450.

Per Martin-Löf and Giovanni Sambin. 1984. *Intuitionistic type theory*. Vol. 9. Bibliopolis Naples.

Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

Ulf Norell. 2008. Dependently typed programming in Agda. In *International School on Advanced Functional Programming*. Springer, 230–266.

Pablo Lamela Seijas, Simon J Thompson, and Darryl McAdams. 2016. Scripting smart contracts for distributed ledger technology. *IACR Cryptology ePrint Archive* 2016 (2016), 1156.

Ilya Sergey, Amrit Kumar, and Aquinas Hobor. 2018. Scilla: a smart contract intermediate-level language. *arXiv preprint arXiv:1801.00687* (2018).

Anton Setzer. 2018. Modelling Bitcoin in Agda. *arXiv preprint arXiv:1804.06398* (2018).

Joachim Zahnentferner. 2018. An Abstract Model of UTxO-based Cryptocurrencies with Scripts. *IACR Cryptology ePrint Archive* 2018 (2018), 469.

Joachim Zahnentferner and Input Output HK. 2018. *Chimeric ledgers: Translating and unifying utxo-based and account-based cryptocurrencies*. Technical Report. Cryptology ePrint Archive, Report 2018/262, 2018. https://epri nt. iacr. org ….