Streamlet in EPR Using IVy

Sydney Gibson, Afonso Tinoco, Mihir Bala, and Ke Wu

1 Introduction

Distributed consensus protocols have become a staple of today's cryptocurrency landscape. In theory, they enable multiple geographically scattered machines to agree upon a transaction, despite unreliable communication and Byzantine failures. In practice, however, they are difficult to implement correctly due to widespread extrinsic non-determinism. Despite these issues, making strong guarantees about these protocols might be possible. In recent years, formal verification tools have become popular, enabling users to rule out large classes of implementation bugs and prove useful properties about their code. Although they have seen limited application in distributed systems contexts, some researchers have already begun experimenting with them on well-known consensus protocols [11]. In this paper, we discuss our journey in implementing Streamlet [1], a blockchain-based consensus protocol, using formal verification tools. In doing so, we will highlight the successes and failures we encountered along the way while also discussing the abilities and limitations of current verification tools.

2 Background and Motivation

Distributed consensus protocol implementations are notoriously difficult to debug. This is due to their reliance on asynchronous, unreliable network code along with their potential for non-determinism. Unfortunately, with their use in popular cryptocurrencies like Bitcoin and Etherium, it has become increasingly critical that such protocols are implemented correctly. Failures could result in fraudulent transactions, causing serious financial damage and compromising user security. In late 2020, a memory out-of-bounds bug was found in the Etherium code base, causing a fork. Luckily, no user data was compromised. A verified consensus protocol implementation, if executed correctly, could guarantee that such bugs will never arise, a feature which would not only increase the system's integrity but also bolster user confidence. This is not a simple process, however. Network code is very difficult to verify which poses serious problems when designing a verified distributed program. Thus, we settled on a relatively simple blockchain consensus protocol, Streamlet. For our implementation, we decided to use Microsoft IVy, a verification language and framework for automatic verification designed with distributed protocols in mind.

2.1 Primer on IVy

IVy is a protocol verification system and programming language built to specify, implement and verify protocols described in effectively propositional logic (EPR). EPR is a decidable subset of first order logic that contains formulas with quantifiers only as prefixes and without uninterpreted functions.

IVy was developed with the goal of making system verification convenient and to minimize proof writing effort. Because EPR is a decidable logic, as long as the developers can describe the system in EPR, an automated theorem prover (in the case of IVy, Z3[2]) can always either prove invariants about the system, or come up with counterexamples along with suggested stronger invariants. IVy also supports full first order logic, but the automated theorem prover might not terminate in that case.

Compared to Linear Temporal Logic (LTL) verification frameworks, such as TLA$^+$[4], IVy does not have a built-in way to specify eventual temporal properties. However, it does allow users to specify atomic state-changing actions and invariants maintained between those actions. Therefore, in order to describe eventual temporal properties in IVy, a user must provide bounds for the eventual properties (such as liveness). This is a double edged sword, as, although it enforces users to write more invariants that could otherwise be omitted