WIP: Formalizing the Concordium consensus protocol in Coq

Thomas Dinsdale-Young
Concordium ApS, Aarhus

Søren Eller Thomsen
Computer Science, Aarhus University

DIGIT

Daniel Tschudi
Computer Science, Aarhus University

ACM Reference Format:

We report an industrial application of Coq: our work towards formalizing the Concordium blockchain consensus protocol. This work is currently under NDA, but we expect to be able to release more information by the time of CoqPL. Eventually, all the code will be published under a permissive open source license. This work is a collaboration between Concordium and Aarhus University.

The Concordium consensus protocol is a provably secure Proof-of-Stake protocol (PoS). The protocol has a prototype implementation in Haskell. Our aim is to connect the cryptographic security proof with the Haskell implementation. Ideally, we would either use Coq’s extraction mechanism or hs-to-coq [SBRW17] to connect the formalization with the Haskell implementation. Our work is loosely based on the toychain formalization in Coq [PS18]. Toychain is a formalization of a modified bitcoin-like Proof-of-Work (PoW) protocol. Like toychain, we build on the math-components library.

Formalizing a PoS protocol is inherently different from formalizing a PoW protocol. In PoW a block-correctness-proof not fixed to a specific block, but rather to a specific slot. Based on [BMTZ17] we do however aim to find a good modularization of the protocol. This will allow for a common generalization of both PoS and PoW protocols.

In our Coq formalization, we abstract away from the P2P-layer and formalize an ideal semi-synchronous network functionality. In toychain every peer relays a message containing hashes of its current stash of blocks to all peers, each time the peer receives a block. We verify a protocol without this message redundancy. At the workshop we will present the Coq implementation of the protocol and its functional correctness.

To prove security, we need to add probabilistic computation. This allows us to add cryptographic primitives such as hash functions. To do so, we will use a probabilistic library in Coq: either the one from information theory [AG15], following a successor of toychain [GS18], or the classical analysis library in math-components which has been used to formalize the logic underlying easyecrypt.

Next we will need to reason about adversaries. This will most likely combine ideas from the formalization of multiparty computation in easyecrypt [HKO*18, ABB*18] and insights from abstract cryptographic frameworks [BDLKK18, Mau11] and the rational analysis of Bitcoin [BGM*18].

Finally, to support that using extracted code can be efficient, we mention the impressive work for the Raft (permissioned) consensus protocol which is formalized in Coq using the Verdi-framework [WWP*15, WWA*16]. It proves that the raft protocol is linearizable, and extracts an efficient implementation of the protocol with performance comparable to a state of the art implementation. A small trusted shim connects the extracted functional code to the running distributed code.

Concordium aims to collaborate with other science heavy companies in setting up a collaborative effort to develop clear quality standards for blockchains. At the end of this lecture, we invite discussion on how Coq is best used in this process.

Acknowledgments
We would like to thank Elaine Li who was involved in the beginning of this work. Ilya Sergey has been very helpful in discussing their upcoming work on formalizing the bitcoin backbone protocol. This work has been supported by the Concordium Blockchain Research Center, Aarhus University, Denmark.
References


