Verifying Finality for Blockchain Systems

Karl Palmskog Milos Gligoric
University of Texas at Austin, USA
palmskog@utexas.edu,gligoric@utexas.edu

Lucas Peña Grigore Roşu
University of Illinois at Urbana-Champaign, USA
lpena7@illinois.edu,grosu@illinois.edu

1 Introduction

Blockchain systems such as Bitcoin [12] and Ethereum [14] are increasingly used as financial transaction mechanisms (cryptocurrencies). A desirable property of a transaction mechanism is *durability*—after a user has submitted a transaction and received initial confirmation, the transaction should not be rolled back. However, in a blockchain system, there may be several competing chains of blocks that agree on the transaction history from the initial *genesis block* only up to some point. These *forks* may arise as a result of network delays or adversarial behavior by some nodes, and can lead to transactions disappearing as soon as one of the forks is preferred by most nodes. To address the problem of long-ranging blockchain revisions, Buterin and Griffith proposed Casper [9], a *finality* system that overlays a proof-of-work blockchain such as Ethereum. When a large enough fraction of participants in the system are honest, Casper defends both against active attacks and catastrophic crashes.

In ongoing work, we are formally verifying Casper in Coq, both at the abstract protocol level and at the level of a distributed blockchain system. In the terminology of Appel et al. [7], we aim to make the Casper specification *two-sided*: implementable in Ethereum nodes and provably beneficial to Ethereum users. A further goal is to lay the foundation for making the Casper specification *live*, i.e., enable verifying Ethereum node implementations. Below, we give some background on Casper and previous verification efforts, and then describe our modeling and verification approach.

2 Background

We build directly on two previous modeling and verification efforts: verified abstract models of various versions of Casper in Isabelle/HOL by Hirai [11], and a model in Coq of a distributed blockchain system called Toychain by Pîrlea and Sergey [13]. We give a brief overview of the Casper finality system and each of these two pillars.

Casper finality. The global state in a blockchain system can be viewed as consisting of a *block tree*, with the genesis block at the root. New blocks with transactions are continually being added to the tree through a *proposal* mechanism such as proof-of-work [12]. Intuitively, Casper works by engaging a group of *validators* who then attest to, by broadcasted votes, that certain blocks in the tree belong to a designated canonical blockchain. When a certain block has been attested to by enough validators, it is deemed to be *finalized*. To participate, validators are forced to demonstrate that they have a stake in the blockchain system by making a cryptocurrency deposit. The deposit may be *slashed* if the validator is verifiably reported by other validators to be behaving adversarially; the reporting validators are then rewarded.

For verification, we focus on two properties of Casper proved informally [9]: accountable safety and plausible liveness. Accountable safety states that conflicting blocks in different block tree forks cannot both be finalized if more than $\frac{2}{3}$ of validators *by deposit* behave honestly. Plausible liveness states that regardless of what has happened before, it is always possible to continue to finalize blocks when more than $\frac{2}{3}$ of validators by deposit follow the protocol.

Casper formalizations. Hirai formalized and verified several variants of Casper in Isabelle/HOL [11]. These formalizations are highly abstract, in the sense that they elide most details on the structure of hashes, blocks, and validators. For example, the requirements on the fractions of validators is captured, via Isabelle's locale mechanism [8], as a constraint on membership in abstract sets:

$$\bigwedge q_1\ q_2\ .\ \exists\ q_3\ .\ \forall\ v\ .\ v \in_2 q_3 \rightarrow v \in_1 q_1 \wedge v \in_1 q_2$$

Here, $\bigwedge$ is all-quantification, and $v \in_i q$ means that the validator v belongs to a set q drawn from some validator powerset (quorum) identified by i. While accountable safety is verified for the most recent Casper, plausible liveness is only proven for an earlier Casper version with different inter-validator messages.

Toychain. Toychain is a formalization of blockchain systems in Coq using the MathComp library [4]. It defines blocks, forks, and distributed node state, but abstracts from specific block proposal mechanisms. Toychain describes the behavior of a blockchain system as a relation between global states, and establishes that absent adversarial interference, a canonical chain becomes known to all nodes in the steady state. For example, the global state is a Coq record

```
Record World := mkW { localState : StateMap;
 inFlightMsgs : seq Packet; consumedMsgs : seq Packet; }.
```

where `localState` maps node names to their current block tree and other local data. We have extended and revised Toychain in collaboration with its authors to support capturing full realistic blockchain system specifications such as that for Bitcoin [2]. Our distributed system model for Casper verification in Coq is derived from this version of Toychain.

CoqPL'19, January 19, 2019, Lisbon, Portugal
2019.

3 Modeling and Verification Approach

We decided to translate Hirai's Casper definitions and theorems [11] from Isabelle/HOL to Coq, and connect the resulting Casper definitions with the Toychain definitions. At the same time, we are extending the Toychain distributed system definitions to capture the behavior of nodes as found in the Casper-based beacon chain for Ethereum [1].

From Isabelle/HOL to Coq. We initially focused on the accountable safety property. Since the development for safety by Hirai mostly uses first-order reasoning, we were able to successfully leverage the CoqHammer extension [3, 10] to perform proofs in Coq that closely followed Isabelle/HOL proofs. At the same time, we reformulated Isabelle locale variables to Coq section variables using MathComp concepts, with validators as a finite type `V`:

```
Variables quorum_1 quorum_2 : {set {set V}}.
Hypothesis qs : ∀ q1 q2, q1 ∈ quorum_1 → q2 ∈ quorum_1 →
 ∃ q3, q3 ∈ quorum_2 ∧ q3 \subset q1 ∧ q3 \subset q2.
```

Defining a global state `s` as the votes cast by validators on block hashes, this hypothesis (along with suitable assumptions on the block parent hash relation) allowed us to prove the following property for arbitrary states:

```
fork s → ∃ q, q ∈ quorum_2 ∧ ∀ v, v ∈ q → slashed s v
```

Intuitively, this property states that if a fork arises from validator attestations, some set of validators in `quorum_2` have misbehaved and will have their deposits slashed. Abstractly, this corresponds to the informal notion of accountable safety.

Bridging the gap to Toychain. Under cursory inspection, the definitions and proven properties in the abstract models of Casper are quite far from those in the Casper paper [9]. Our goal is to instantiate the properties in the more concrete setting of Toychain and bring them closer to the pen-and-paper versions. Assuming deposits are given by a function `d : V → nat`, we can instantiate the hypothesis `qs` from above by reasoning on powersets of validators that have combined deposits equal to or above some amount `n`:

```
[set x in powerset [set: V] | \sum_(v in x) (d v) >= n]
```

For example, `quorum_1` can be defined along these lines with `((2 * \sum_(v : V) (d v)) %/ 3).+1` for `n`, and `quorum_2` can be defined with `((\sum_(v : V) (d v)) %/ 3).+1` for `n`, consistent with the informal definitions.

Working from the other end of the abstraction spectrum, we define functions and datatypes in Coq following those in the beacon chain implementation [1], to be used when defining Toychain node state and node-local behavior. For example, following Toychain, a block can be abstracted to a Coq record type containing, most notably, a hash of the previous block and a collection of attestations by validators:

```
Record Block := mkB { parent_hash : Hash;
  attestations : seq Attestation;
  (* ... omitted fields ... *) }.
```

Using notions from the FCSL PCM library [5], we then define block trees, which are part of the state of each distributed node, as finite maps from hashes to blocks:

```
Definition Blocktree := union_map Hash Block.
```

This allows us to instantiate a suitably concrete parent relation over hashes:

```
Definition hash_parent (bt : Blocktree) : rel Hash :=
[rel x y | (x ∈ dom bt) && if find y bt is Some b
 then parent_hash b == x else false].
```

Ongoing refinements include aligning the Toychain concept of a fork, which is in terms of block sequence prefixing, with the more abstract Casper notion of a fork in terms of finalized blocks over the closure of the parent hash relation.

Ultimately, we aim to transfer accountable safety and plausible liveness from Hirai's abstract models to hold for steps over the Toychain global state according to a step relation and node state definition matching those in the Ethereum reference beacon chain node implementation [1]. That is, we will express and prove both properties along the lines of the *clique consensus* property in Toychain [13], i.e., similarly to

```
Theorem accountable_safety_inv : ∀ (w w' : World),
 accountable_safety w → step w w' → accountable_safety w'.
```

We then plan to use the step relation as a basis for a verified node implementation. We also want to capture *sharded* systems, which consist of many separate blockchains for reasons of scalability [6], and to keep up with the Casper protocol design, which is a moving target.

References

[1] 2018. Beacon Chain. https://github.com/ethereum/beacon_chain/
[2] 2018. Bitoychain. https://github.com/palmskog/bitoychain
[3] 2018. CoqHammer. https://github.com/lukaszcz/coqhammer
[4] 2018. Mathematical Components Project. https://math-comp.github.io/math-comp/
[5] 2018. PCM library. https://github.com/imdea-software/fcsl-pcm
[6] 2018. Sharding FAQs. https://github.com/ethereum/wiki/wiki/Sharding-FAQs
[7] Andrew W. Appel, Lennart Beringer, Adam Chlipala, Benjamin C. Pierce, Zhong Shao, Stephanie Weirich, and Steve Zdancewic. 2017. Position paper: the science of deep specification. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 375, 2104 (2017).
[8] Clemens Ballarin. 2014. Locales: A Module System for Mathematical Theories. *Journal of Automated Reasoning* 52, 2 (01 Feb 2014), 123–153.
[9] Vitalik Buterin and Virgil Griffith. 2017. Casper the Friendly Finality Gadget. *CoRR* abs/1710.09437 (2017).
[10] Łukasz Czajka and Cezary Kaliszyk. 2018. Hammer for Coq: Automation for Dependent Type Theory. *Journal of Automated Reasoning* 61, 1 (2018), 423–453.
[11] Yoichi Hirai. 2018. A repository for PoS related formal methods. https://github.com/palmskog/pos
[12] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/bitcoin.pdf
[13] George Pîrlea and Ilya Sergey. 2018. Mechanising blockchain consensus. In *Certified Programs and Proofs*. 78–90.
[14] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. (2014). http://gavwood.com/paper.pdf