On the Formal Verification of the Stellar Consensus Protocol

Giuliano Losa
Galois, Inc., Portland, Oregon, USA
giuliano@galois.com

Mike Dodds
Galois, Inc., Portland, Oregon, USA
miked@galois.com

Abstract

The Stellar Consensus Protocol (SCP) is a quorum-based BFT consensus protocol. However, instead of using threshold-based quorums, SCP is permissionless and its quorum system emerges from participants' self-declared trust relationships. In this paper, we describe the methodology we deploy to formally verify the safety and liveness of SCP for arbitrary but fixed configurations.

The proof uses a combination of Ivy and Isabelle/HOL. In Ivy, we model SCP in first-order logic, and we verify safety and liveness under eventual synchrony. In Isabelle/HOL, we prove the validity of our first-order encoding with respect to a more direct higher-order model. SCP is currently deployed in the Stellar Network, and we believe this is the first mechanized proof of both safety and liveness, specified in LTL, for a deployed BFT protocol.

2012 ACM Subject Classification Software and its engineering → Software verification; Networks → Protocol testing and verification

Keywords and phrases Consensus, Blockchains, First-Order Logic, Stellar, Ivy Prover, Decidability

Digital Object Identifier 10.4230/OASIcs.FMBC.2020.9

Funding This work was supported by the Stellar Development Foundation.

1 Introduction

Blockchains rely on Byzantine Fault-Tolerant (abbreviated BFT) consensus protocols to ensure that, despite the presence of malicious participants, the network of participants as a whole eventually reaches consensus on what block to append next to the blockchain. In many blockchains, the security of large amount of digital assets depend on the correctness of the blockchain's BFT consensus protocol, but designing BFT consensus protocols is notoriously difficult and serious flaws can remain undetected for years [1].

While formal verification can prevent many correctness issues in BFT consensus protocols, performing such verification is challenging for several reasons: BFT consensus protocols are designed to support an arbitrary number of participants; their executions and their reachable state-space are unbounded; they operate in asynchronous networks where the interleaving of messages is unpredictable; and finally, verifying termination is as important as verifying that participants never disagree.

In this paper, we summarize our approach to the formal verification of the main safety and liveness properties of the Stellar Consensus Protocol (abbreviated SCP) [7] using the Ivy methodology [10]. Both the safety and liveness proofs apply to a unique model of SCP. This model is parameterized by a fixed but arbitrary set of participants and denotes a set of infinite executions. To our knowledge, this is the first work that mechanically proves both safety and liveness, expressed in LTL, of a deployed BFT protocol under arbitrary configurations.

2nd Workshop on Formal Methods for Blockchains (FMBC 2020).
Editors: Bruno Bernardo and Diego Marmsoler; Article No. 9; pp. 9:1–9:9
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

At a high level, verifying the safety of a protocol with Ivy entails 1) developing a set of axioms to express the protocol's underlying domain model as a first-order theory over uninterpreted sorts; 2) modeling the protocol in Ivy's procedural language; 3) developing an inductive invariant that implies the safety properties, while ensuring that verification conditions fall into the decidable first-order logic fragment EPR [5]. This is facilitated by Ivy's modular decomposition features [17].

For termination, or more generally liveness, Ivy provides a liveness to safety reduction [12] crafted specifically to help produce decidable verification conditions. Given a temporal property in First-Order Linear Temporal Logic (FO-LTL), Ivy automatically synthesises a transition system and an associated safety property such that if the synthesized system is safe, then the temporal property of the original system holds. The user can then verify that the synthesized transition system is safe using the safety verification methodology.

Producing EPR verification conditions ensures that Z3 can automatically and reliably determine their satisfiability. Compared to approaches that use automation but do not require decidability, Ivy's predictable automation greatly simplifies the mental model of the prover that the user must keep in mind when developing a proof. The user can thus stop worrying about the prover and instead focus on the properties of the protocol.

A key challenge in applying the Ivy methodology to SCP is to model SCP's permissionless Federated Byzantine Quorum Systems in first-order logic and in a way that is conductive to decidable reasoning in EPR. In SCP, every participant expresses agreement requirements with other nodes, and SCP relies on the properties of the resulting graph-like structure to solve consensus. At first sight, such a complex family of structures seem hard to axiomatize in first-order logic, let alone EPR.

The rest of the paper focuses on the first-order logic modeling of Federated Byzantine Quorum Systems. This model abstracts over significant aspects of SCP's quorum system. To provide evidence that the abstraction is sound, we verify some of its key properties with respect to a more concrete model in Isabelle/HOL.

With a first-order theory of Federated Byzantine Quorum Systems established, we verify that SCP's balloting protocol [7] satisfies its agreement property and that, under eventual synchrony, it satisfies its termination property (i.e. that every node eventually decides). This safety and liveness proof largely follows patterns identified previously during the verification of other consensus protocols in Ivy [14, 12, 3], and it is not described in this paper.

Our paper supplies evidence that BFT consensus protocol can be verified with decidable logics, which enables powerful yet stable automation, using the Ivy methodology. The proof is available online [8], and, for a more complete reference, we plan to publish an extended version of this paper with a detailed account of the safety and liveness proof that are omitted here.

2 Solving Consensus in a Federated Byzantine Quorum System

SCP must solve consensus, guaranteeing agreement and termination, in a permissionless system where nodes can join or leave without any synchronization and without the permission of any gatekeeper. There is thus no common notion of the set of all nodes. Moreover, the system is susceptible to Sybil attacks, in which attackers create a large number of identities to try to overwhelm the system. In such an environment, traditional threshold-based quorum systems, defined in terms of the total number of nodes, are thus of no use.

Other permissionless protocols like Bitcoin or Algorand use Proof-of-Work or Proof-of-Stake to defend against Sybil attacks. Stellar takes a different approach. The Stellar Network is intended as a platform to exchange digitized real-world assets (e.g. land parcels, retail coupons, national currencies, agricultural goods, etc.). Most participants are thus expected to engage with recognized identities and have real-world relationships with some (but not all) other participants in the network. SCP leverages these real-world relationships to defend against Sybil attacks, counting on real-world relationships to be difficult for an attacker to establish.

Concretely, each node in the Stellar Network is required to independently declare a set of *slices*, where each slice is a set of nodes. The intent is that a node n accepts some new information it hears on the network if and only if one of its slices unanimously agrees that the information is correct. Thus slices can be thought of as agreement requirements. Nodes advertise their slices throughout the network, and each node forms its own, personal notion of quorum based on its own slices and on the slices of other nodes it knows about, as follows. A quorum of n is defined as a set Q of nodes such that a) n has a slice included in Q and b) each member of Q has a slice included in Q. In other words, a quorum of n is a set that n satisfies the agreement requirement of n and of all its members. Formally, let $\text{slices}(n)$ denote the set of slices of node n. Then a set of nodes Q is a quorum of a node n if a) $\exists S \in \text{slices}(n).\ S \subseteq Q$ and b) $\forall m \in Q.\ \exists S \in \text{slices}(m).\ S \subseteq Q$. The resulting quorum system is called a Federated Byzantine Quorum System (abbreviated FBQS).

2.1 Intact and Intertwined Sets

With the notion of quorum in place, it seems possible to take a traditional threshold-based BFT consensus protocols, and only change how quorums are defined in order to obtain a consensus protocol for the Stellar Network. However, FBQS have some unusual properties that complicate the task. First, the notion of quorum is not global to the system; instead, each node has its own view of what a quorum is. Second, the quorums of a node depend on what slices other nodes declare; thus, Byzantine nodes can influence a well-behaved node's notion of what a quorum is. Third, it is possible that a subset of the nodes have quorums that intersect enough to guarantee safety, while some other subsets do not; thus, consensus may be solvable for only a strict subset of the system; there may even by two or more disjoint subsets of the system that form consensus islands that nevertheless diverge from each other.

What properties must a set of nodes satisfy in order for consensus to be solvable among its members? We do not know a precise answer to this question [9]. However, we can prove that SCP solves eventually synchronous consensus among sets of nodes called *intact sets*. A set I of well-behaved (non-Byzantine) nodes is intact when, regardless of what slices Byzantine nodes advertise: a) I enjoys quorum availability, i.e. the set I is a quorum for all its members, and b) I enjoys quorum intersection, i.e. if n_1 and n_2 are members of I, if Q_1 is a quorum of n_1, and if and Q_2 is a quorum of n_2, then the intersection of Q_1 and Q_2 contains a member of I. An important property of intact sets is that the union of two intact sects that intersect is also an intact set; thus maximal intact sets are disjoint and form consensus islands within the network.

Sets which have quorum intersection but which lack quorum availability are called *intertwined* sets. Precisely, a set S of nodes is intertwined when, if n_1 and n_2 are members of S, if Q_1 is a quorum of n_1, and if Q_2 is a quorum of n_2, then the intersection of Q_1 and Q_2 contains an intertwined member. SCP also guarantees that there will not be any disagreement among an *intertwined* set.

2.2 Termination and the Cascade Theorem

Thanks to the quorum intersection property, it is easy to guarantee agreement to an intertwined set. However, termination is more difficult to achieve. Traditional BFT consensus protocols often rely on eventual synchrony [4] to ensure termination. The idea is that, once the system becomes synchronous, the protocol can rely on all nodes having the same view of the system.

For example, suppose that, in a threshold quorum system, a quorum Q unanimously agrees on statement X. If the network is synchronous, then all nodes shortly notice that Q unanimously agrees on X. In this sense, they all form the same view of the fact "there is a quorum that is unanimous about X". Instead, if the quorum system is not a threshold quorum system but an FBQS, then no such common view arises because Q may be a quorum only of some nodes but not others.

SCP circumvents this problem using an epidemic propagation phenomenon that guarantees that, once the system is synchronous, if an intact node witnesses a unanimous quorum, then the knowledge that there is such a quorum soon propagates to the entire intact set, and Byzantine nodes cannot prevent propagation.

The epidemic propagation phenomenon is enabled by the Cascade Theorem. This theorem relies on the notion of slice-blocking set. A set B is a slice-blocking set for a node n when every slice of n intersects B. The cascading theorem states that if n is intact, Q is a quorum of n, and U is a superset of Q, then either all intact nodes belong to U, or U slice-blocks some intact node $m \notin U$.

Let us now get back to the example in which a quorum Q of an intact node unanimously agrees on statement X. We would like that all intact nodes learn the fact "there exists a quorum of an intact node that unanimously agrees on X". By the Cascade Theorem, either all intact nodes already know the fact, or there must be an intact node n that does not know it but that is slice-blocked by a set of intact nodes that know it. Thus, if we add the rule that n must accept a fact if slice-blocked by a set that already accepted the fact, then n newly accepts the fact. This process then repeats until the knowledge of the existence of Q propagates to the entire intact set.

Finally, we must also be sure that malicious nodes cannot use epidemic propagation to propagate forged facts. This is guaranteed because if n is intact and S slice-blocks n, then S contains an intact node.

3 Modeling Federated Byzantine Quorum Systems in EPR

In this section, we describe the first-order theory of Federated Byzantine Quorum System. This is the model we use in our proofs of safety and liveness (the proofs themselves are not described in detail in this paper).

3.1 Enabling Decidable Reasoning

We craft the FBQS model to meet two constraints: on the one hand, the model must enable decidable automated reasoning in EPR; on the other hand, the model must accurately capture the properties that FBQSs have in practice. Our solution is to abstract over some important aspects of FBQSs to make decidable reasoning possible, while formally verifying that the model is sound with respect to a more concrete model developed in Isabelle/HOL. By doing so, we trade off a relatively small manual proof effort in Isabelle/HOL in exchange for decidable automated reasoning in Ivy.

To enable decidable reasoning with Ivy, we model FBQSs as a first-order theory consisting of: a) a set of uninterpreted sorts, b) constants, functions, and relations over those sorts, and c) first-order axioms that constrain the models of the theory to structures that have properties sufficient for the balloting protocol to be correct. Moreover, we must use quantifier alternations and functions carefully, as those will impact our ability to keep protocol verification conditions in EPR.

A verification condition is in EPR when its sorts are stratified: for every pair of sorts a and b, say that b depends on a if either a) an existential quantifier on sort b is in the scope of a universal quantifier on sort a, or b) there is a function symbol of type $a_1, \cdots, a_n \to b$ with $a = a_j$ for some $j \in 1 \cdots n$; sorts are stratified if the dependencies between sort do not form any loops or cycles. For example, in the formula $\forall x.\exists y.P(x, y)$ where x is of sort a and y is of sort b and P is a predicate symbol, sort b depends on sort a but the formula is stratified. However, if both x and y have the same sort, then there is a sort dependency loop and the formula is not stratified.

Protocol verification conditions are formulas of the form $A \wedge I \wedge T \wedge \neg I'$, where A is the conjunction of the FBQS theory axioms, I is a protocol invariant, T is the protocol's transition relation, and I' is the post-state version of I. Thus unstratified verification conditions can arise because of the interaction between axioms, invariants, their negation, and the protocol's transition relation. It is thus wise to minimize the use of function symbols and quantifier alternation when developing the EPR FBQS theory.

In our experience, stratification is nevertheless likely to become an issue during protocol verification. However, Ivy has modular decomposition features specifically designed to help keep verification conditions decidable. The process of structuring proof modularly to ensure decidability is explained in details by Taube et al. [17]. In the case of liveness proofs, prophecy variables also help keep verification conditions decidable [13].

3.2 The Unique Challenges Posed by FBQSs

Developing an EPR theory of FBQSs is challenging because the notions we presented in Section 2, such as intact sets, slice-blocking sets, or the cascading theorem, are naturally second-order concepts. I.e. they are naturally expressed by quantifying over sets. While we cannot precisely capture the higher-order theory of sets in first-order logic, we can approximate it by using a first-order uninterpreted sort `nset`, a membership relation `member(N:node, S:nset)`, and appropriate axioms.

The full first-order theory of FBQSs appears in Figure 1. We model an arbitrary but fixed configuration, i.e. an arbitrary set of nodes with arbitrary slices and we consider a fixed intact set I among those nodes as well as a superset S of I such that S is intertwined (note that an intact set is inherently intertwined, so there is no inconsistency here). Instead of modeling slices explicitly, we only model the notions of intact node, intertwined node, quorum, and slice-blocking set.

Formally, in Figure 1, we introduce an uninterpreted sort `node`, denoting the set of all nodes, and an uninterpreted sort `nset`, denoting the powerset of the set of nodes (lines 1 and 2). Well-behaved, intertwined, and intact nodes are identified by corresponding unary relations (lines 3 to 5), and quorums of a node are identified by the binary relation `quorum_of` (line 7). Finally, the binary relation `member` (line 6) denotes set membership, and the binary relation `slice_blocking` identifies the slice-blocking sets of a node (line 8).

Given those sorts and relations, we obtain the first-order theory of FBQSs using the following axioms. First, line 9, we assert that intact nodes are intertwined, and that intertwined nodes are well-behaved. In line 10 and 11, we assert that quorums of well-behaved nodes are not empty. Then, line 12 to 15, we define two predicates to identify quorums of intertwined nodes and quorums of intact nodes. Then, line 16 and 17, we assert the quorum intersection property of intertwined nodes. Similarly, line 18 and 19, we assert the quorum intersection property of intact nodes. Line 20 and 21, we assert that if `N` is intact and `S` slice-blocks `N`, then `S` contains an intact node. Finally, line 22, we assert that the set of intact nodes is a quorum.

The conjunction of all the axioms is an EPR formula because the associated quantifier-alternation graph has a single dependency: sort `node` depends on sort `nset`. For example, lines 16 and 17 in Figure 1, the quorum intersection axiom for intertwined sets creates a dependency from sort `node` to sort `nset`. As explained in Section 3.1, this dependency may create a quantifier-alternation cycle when the axioms are conjoined with other formulas in a verification condition, and it is the user's responsibility to make use of Ivy's modularity features to avoid such a cycle when verifying a protocol; this process is explained in [17].

The reader may notice that the Cascade Theorem is missing from the axioms, and instead is expressed as an axiom schema in Figure 2. The reason is that we could not satisfactorily express it in first-order logic. The theorem states that if `p` is a predicate on nodes (i.e. a set of nodes) and `Q` is a quorum of an intact node whose intact members unanimously satisfy `p`, then either a) all intact nodes satisfy `p` or b) there exists an intact node `N` that does not satisfy `p` but that is slice-blocked by a set `S` whose members are exclusively intact and unanimously satisfy `p`. While other axioms quantify over a restricted family of sets, such as quorums or slice-blocking sets, the Cascade Theorem quantifies over all predicates `p`. It is thus inherently second-order. Ivy allows to express it as an axiom schema, but Ivy's proof automation cannot reason about such a second-order formula. Instead, Ivy allows to manually instantiate it, substituting `p` for a concrete predicate, to prove particular invariants. We use this technique in the termination proof of SCP. Note that, also when instantiating the Cascade Theorem, we must be careful not to introduce quantifier-alternation cycles.

Together, the axioms appearing in Figure 1 and the axiom schema of Figure 2 form the first-order theory of Federated Byzantine Quorum Systems.

3.3 Validating the Model

Asserting axioms instead of proving them as properties from basic definitions can be dangerous: even benign-looking axioms can turn out to be contradictory, e.g. because of a typo, thereby making any proof relying on them vacuous. To avoid this situation, we ask Ivy to find a model of the axioms of Figure 1 conjoined with the instantiations of the `cascade_thm` axiom schema that we use in the proof. Ivy confirms the existence of a model, which rules out any contradiction.

Another risk is that, although the axioms are not contradictory, they do not accurately model FBQSs. For instance, the first-order model abstracts over slices and instead considers that a node's quorums are fixed. This is limiting because, in reality, nodes are expected to change their slices in response to observed failures or changes in how much they trust other nodes. It is nevertheless interesting to prove that, under the assumption that well-behaved nodes do not change their slices, SCP is safe and live.

Another issue is that, as we have noted in Section 2, FBQSs have the peculiar property that, by crafting the slices they advertise, malicious nodes can dynamically influence a well-behaved node's notion of quorum. But in our model, the quorums of a well-behaved node are fixed. This is in fact a form of abstraction. Given a FBQS where well-behaved nodes have fixed slices and Byzantine nodes can advertise arbitrary slices, we defined its fixed-quorums counterpart, where we assign to each well-behaved node the set of all quorums that could

```
type node # the type of nodes; this is an uninterpreted, arbitrary non-empty type
type nset # the type of node sets
relation well_behaved(N:node)
relation intertwined(N:node)
relation intact(N:node)
relation member(N:node, S:nset) # this is the set membership relation
relation quorum_of(N:node, Q:nset)
relation slice_blocking(S:nset, N:node)
axiom ∀ N. (intact(N) → intertwined(N)) ∧ (intertwined(N) → well_behaved(N))
axiom (∃ N . well_behaved(N) ∧ quorum_of(N,Q))
    → ∃ N . well_behaved(N) ∧ member(N,Q)
definition quorum_of_intertwined(Q) =
  (∃ N. intertwined(N) ∧ quorum_of(N,Q))
definition quorum_of_intact(Q) =
 (∃ N. intact(N) ∧ quorum_of(N,Q))
axiom ∀ Q1,Q2. quorum_of_intertwined(Q1) ∧ quorum_of_intertwined(Q2)
  → ∃ N. intertwined(N) ∧ member(N,Q1) ∧ member(N,Q2)
axiom ∀ Q1,Q2. quorum_of_intact(Q1) ∧ quorum_of_intact(Q2)
  → ∃ N. intact(N) ∧ member(N,Q1) ∧ member(N,Q2)
axiom ∀ S. (∃ N. intact(N) ∧ slice_blocking(S,N))
  → ∃ N2. member(N2,S) ∧ intact(N2)
axiom ∃ Q. ∀ N. member(N,Q) ↔ intact(N) ∧ quorum_of(N,Q)
```

Figure 1 A model of Federated Byzantine Quorum Systems in the EPR fragment of first-order logic.

possibly arise given arbitrary malicious behavior of Byzantine nodes. Intuitively, in this new fixed-quorums model, it is harder to achieve consensus because the Byzantine adversary has more choices of quorums to manipulate. Thus a consensus algorithm that works in the fixed-quorums model will work in the model in which quorums can be dynamically shaped by Byzantine nodes.

Finally, we prove in Isabelle/HOL that the fixed-quorums model satisfies all the properties axiomatized in the first-order model; thus the first-order model is an abstraction of the fixed-quorums model. We now describe the Isabelle/HOL fixed-quorums model.

The Isabelle/HOL model formalizes FBQSs from the notion of slice. It assumes that well-behaved nodes have fixed slices, but it accounts for the situation in which malicious nodes dynamically shape the quorums of well-behaved nodes. To do so, we define a quorum Q of a node n as a set of nodes such that a) n has a slice included in Q and b) every *well-behaved* member of Q has a slice included in Q. Note how this definition of quorum subtly differs from the one of Section 2. By placing requirements only on well-behaved nodes, we account for any possible slices that could be advertised by malicious nodes. We then prove in Isabelle/HOL that all the axioms of the first-order model (Figure 1) and the Cascade Theorem (Figure 2) hold. This Isabelle/HOL theory is purely definitional (i.e. it does not use axioms).

There is no mechanically-checked connection between Isabelle/HOL and Ivy, and thus the best we can do is to carefully check, by hand, that the Ivy axioms correspond to the properties proved in Isabelle/HOL. Fortunately, the syntax and semantics of first-order formulas in Isabelle/HOL is very close to that of Ivy. This can be seen by comparing the Ivy

```
axiom [cascade_thm] {
  function p(N:node):bool
  property (∃ Q . quorum_of_intact(Q) ∧ (∀ N . intact(N) ∧ member(N,Q) → p(N)))
    → ((∀ N . intact(N) → p(N))
        ∨ (∃ N,S . intact(N) ∧ ¬p(N) ∧ slice_blocking(S,N)
           ∧ (∀ N2 . member(N2,S) → (intact(N2) ∧ p(N2)))))
}
```

Figure 2 The second-order Cascade Theorem as an axiom schema in Ivy.

axiom schema of Figure 2 with its Isabelle/HOL counterpart appearing in Figure 3.

```
theorem cascade:
  fixes P
  assumes "∃ Q . ∃ n . intact n ∧ quorum_of n Q ∧ (∀ n ∈ Q . intact n ⟶ P n)"
  obtains "∀ n . intact n ⟶ P n" | "∃ n S . intact n ∧ ¬P n
    ∧ (∀ S1 ∈ slices n . S ∩ S1 ≠ {}) ∧ (∀ n ∈ S. intact n ∧ P n)"
```

Figure 3 The Cascade Theorem in Isabelle/HOL.

4 Related Work

Lokhava et al [7] discuss the Stellar Network in the broader context of global payments; they also describe at a high level the formal verification effort that is the subject of the present paper. The purpose of the present paper is to dig into the technical details necessary to apply this technique to future proofs of BFT protocols. Losa et al. [9] show that FBQSs are an instance of the more general Personal Byzantine Quorum System model, and we reuse some of the Isabelle/HOL theories developed for this work.

Other works verify safety properties of BFT consensus protocols using Dafny, Coq, or Isabelle/HOL. For example, Alturki et al. verify safety properties of Algorand in Coq [2]. Palmskog et al.[15] and Nakamura et al.[11] verify properties of Ethereum's Casper CBC in Coq and Isabelle/HOL, respectively. Rahli verifies safety properties of PBFT in the Velisarios framework [16], which is based on Coq. IronFleet [6] verifies safety and liveness of a crash-tolerant implementation of Multi-Paxos using Dafny.

Isabelle/HOL, Dafny, and Coq are not restricted by decidable logics, but they lack the specific features that allow Ivy users to restrict verification conditions to a decidable fragment and in turn benefit from reliable proof automation. A series of papers describe the different aspects of decidable reasoning about protocols in Ivy. [14] focuses on modeling and safety verification of consensus protocols at a high level of abstraction. [3] presents a tool to synthesize first-order axioms modeling threshold-based quorum systems.[17] present Ivy's modularity features, which enable decidable safety verification of more complex protocols and their implementations. Finally, Ivy's liveness-to-safety reduction [12] allows decidable reasoning about liveness properties expressed in LTL. Ivy's support for prophecy variables [13] offers an additional tool that helps preserve decidability. In an extended version of this paper, we plan to present the Ivy proofs of safety and liveness of SCP and compare with the works cited above.

References

1 Ittai Abraham, Guy Gueta, Dahlia Malkhi, Lorenzo Alvisi, Rama Kotla, and Jean-Philippe Martin. Revisiting fast practical byzantine fault tolerance. *arXiv preprint arXiv:1712.01367*, 2017.

2 Musab A Alturki, Jing Chen, Victor Luchangco, Brandon Moore, Karl Palmskog, Lucas Peña, and Grigore Roşu. Towards a verified model of the algorand consensus protocol in coq. *arXiv preprint arXiv:1907.05523*, 2019.

3 Idan Berkovits, Marijana Lazić, Giuliano Losa, Oded Padon, and Sharon Shoham. Verification of Threshold-Based Distributed Algorithms by Decomposition to Decidable Logics. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification*, Lecture Notes in Computer Science, pages 245–266, Cham, 2019. Springer International Publishing.

4 Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

5 Yeting Ge and Leonardo De Moura. Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In *Computer Aided Verification*, pages 306–320. Springer, 2009.

6 Chris Hawblitzel, Jon Howell, Manos Kapritsos, Jacob R. Lorch, Bryan Parno, Michael L. Roberts, Srinath Setty, and Brian Zill. IronFleet: Proving Practical Distributed Systems Correct. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, pages 1–17, New York, NY, USA, 2015. ACM.

7 Marta Lokhava, Giuliano Losa, David Mazières, Graydon Hoare, Nicolas Barry, Eli Gafni, Jonathan Jove, Rafał Malinowsky, and Jed McCaleb. Fast and secure global payments with Stellar. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, pages 80–96, New York, NY, USA, October 2019. Association for Computing Machinery.

8 Giuliano Losa. `https://github.com/stellar/scp-proofs`, 2019.

9 Giuliano Losa, Eli Gafni, and David Mazières. Stellar Consensus by Instantiation. In Jukka Suomela, editor, *33rd International Symposium on Distributed Computing (DISC 2019)*, volume 146 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

10 Kenneth L. McMillan and Oded Padon. Deductive verification in decidable fragments with Ivy. In *International Static Analysis Symposium*, pages 43–55. Springer, 2018.

11 Ryuya Nakamura, Takayuki Jimba, and Dominik Harz. Refinement and Verification of CBC Casper. In *2019 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 26–38, June 2019.

12 Oded Padon, Jochen Hoenicke, Giuliano Losa, Andreas Podelski, Mooly Sagiv, and Sharon Shoham. Reducing Liveness to Safety in First-Order Logic. In *45th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2018)*, Los Angeles, 2018.

13 Oded Padon, Jochen Hoenicke, Kenneth L. McMillan, Andreas Podelski, Mooly Sagiv, and Sharon Shoham. Temporal prophecy for proving temporal properties of infinite-state systems. In *2018 Formal Methods in Computer Aided Design (FMCAD)*, pages 1–11. IEEE, 2018.

14 Oded Padon, Giuliano Losa, Mooly Sagiv, and Sharon Shoham. Paxos Made EPR: Decidable Reasoning About Distributed Protocols. *Proc. ACM Program. Lang.*, 1(OOPSLA):108:1–108:31, October 2017.

15 Karl Palmskog, Milos Gligoric, Brandon Moore, Lucas Peña, and Grigore Roşu. Verification of Casper in the Coq Proof Assistant, 2018. URL: `http://hdl.handle.net/2142/102075`.

16 Vincent Rahli, Ivana Vukotic, Marcus Völp, and Paulo Esteves-Verissimo. Velisarios: Byzantine Fault-Tolerant Protocols Powered by Coq. In Amal Ahmed, editor, *Programming Languages and Systems*, Lecture Notes in Computer Science, pages 619–650, Cham, 2018 . Springer International Publishing.

17 Marcelo Taube, Giuliano Losa, Kenneth L. McMillan, Oded Padon, Mooly Sagiv, Sharon Shoham, James R. Wilcox, and Doug Woos. Modularity for decidability of deductive verification with applications to distributed systems. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2018, pages 662–677, New York, NY, USA, June 2018. Association for Computing Machinery.