Consensus, Blockchain and Proof Assistants

Matthew Salazar

December 2018

1 Introduction

For decades, distributed consensus protocols (DCPs) have been of interest to computer scientists, and with the uptrend of blockchain and its applications, DCPs have earned the spotlight of both private, governmental and academic parties. DCPs are notoriously tortuous to reason about, especially when trying to prove that various properties of these protocols hold. Fortunately, proof assistants such as Coq, NuPRL and Isabelle, and SMT provers such as Z3 and Yices have proven to be useful tools for proving these properties because they are correct-by-construction. Some academics have also proposed different frameworks and logics for reasoning about DCPs to increase the efficacy of proof assistants in proving properties about DCPs.

The purpose of this paper is to give a general overview of DCPs and the efforts of using proof assistants for verifying blockchain DCPs. Additionally, this paper will briefly discuss the uses of proof assistants on smart contracts, a blockchain technology. Although the previous efforts noted in this paper are not exhaustive, they effectively highlight the foundations, recent discoveries and intended future work in both the commercial and academic spaces for the use of proof assistants for verifying classical and blockchain DCPs, and verifying smart contracts.

2 DCP Overview

Fundamentally, DCPs have two types of properties that determine their correctness: liveness and safety. Informally, liveness guarantees that "good things", such as the termination of code or agreement of correct processes, will eventually happen. Safety guarantees that "bad things", such as crash or Byzantine faults, do not happen [13].

DCPs such as Paxos, Raft and Two-Phase Commit are assumed to execute in a permissioned setting, where it is necessary to know how many people there are in the protocol execution, and all communication between members in the protocol execution is authenticated.

Permissionless DCPs such as blockchain DCPs use data structures called block forests and relinquish the need to know how many members are in the

protocol execution and the need to authenticate communication between members. Because permissionless DCPs relax the strong assumptions of permissioned DCPs, permissionless DCPs are more flexible and scalable. However, permissionless DCPs typically have a larger energy consumption, and have longer expected liveness properties. It is worth noting that these two categories are not the only settings for DCPs. One such project working at a middle grounds of the permissioned and permissionless setting is Avalanche from Team Rocket [19].

3 Foundations of DCP Verification

Verification of DCPs is critical for many reasons. The main reasons are the reliance of major technology companies on many of these protocols, and the presumed scale of blockchain projects around the world serving as banking or government substitutes. Many of these protocols do not have a central entity to amend these protocols, which typically control precious assets such as personal information or money. Ensuring the correctness of these protocols becomes central to the survival of these protocols in a real-world distributed setting.

Efforts to formally verify DCPs are typically done by proof assistants or SMT solvers. However, SMT solvers usually require bounds on the environment to be used in order to verify correctness. Furthermore, there are projects that integrate SMT solvers into proof assistants, making proof assistants a stronger tool for reasoning about DCPs [4].

Proof assistants like NuPRL and Coq are logical environments based on Constructive Type Theory. The advantage of working with these proof assistants over strategies such as model-checking is that when trying to prove a protocol property, should that property not hold, the environment constructively provides a counterexample, and, should the property hold, the environment provides a formal mapping from the program specification to running code [18]. These constructive features of proof assistants help with reasoning about and managing theorems in complex systems such as DCPs.

Although verification of DCPs has proven nontrivial, there has been much progress in specifying and verifying different DCPs. Notable foundational work on the verification of DCPs includes TLA+, a formal specification language developed by Leslie Lamport that uses set theory to prove safety and temporal logic to prove liveness [14]. Another notable work is the verification of Paxos using EventML in the NuPRL proof assistant. The NuPRL team developed a Logic of Events (LoE) which they implemented as EventML. Using EventML, they proved the correctness of many DCPs including the Paxos protocol [18]. Many other projects have produced results that have significantly progressed verification for DCPs including new specification languages, logics and frameworks. These notable projects are cited in the references [12, 15, 24, 10, 21, 23, 2, 3].

4 Verifying Blockchain DCPs

Because blockchain DCPs relax the permissioned DCP conditions of variable member participation cardinality in protocol execution and free communication between members, there is a disparity between permissioned and permissionless DCPs from a mathematical modelling perspective [16]. However, from a constructive perspective, proving properties such as liveness and safety seem to be quite similar, only requiring reasoning about block forests and some formalized mechanization of blockchain consensus [17].

Further research toward blockchain DCP verification includes developing a model of the network to prove global system safety and eventual consistency in Coq [17], and using probabilistic formal methods in PRISM to verify probabilistic properties of the blockchain DCP [5].

Furthermore, there are a number of commercial projects that address the importance of blockchain DCP verification. Such projects include Hedera Hashgraph, Ethereum Foundation and Tezos. Hedera is the first distributed ledger technology company to prove its asynchronous Byzantine fault tolerance using Coq [9]. Also, the Ethereum Foundation has used Coq to verify properties about its finality system, Casper [6]. Lastly, the Tezos Foundation plans on using Coq to verify its OCaml code for key portions of the Tezos DCP [8].

5 Smart Contracts

A smart contract is a computerized transaction protocol that executes the terms of a contract. In the context of blockchain, these contracts are stored on a distributed ledger and run as scripts on top of a distributed virtual machine that can send or receive assets based on the execution of that contract [22].

Smart contracts are typically short due to the throughput limitations of the current blockchain DCPs, but as these protocols become more scalable, smart contracts have potential to become increasingly large, which complicates the verification of the smart contract for the people entering into this smart contract. Proof assistants emerge as an ideal solution for proving the correctness of smart contracts so that individuals engaging in a smart contract know what the contract does what it is formally specified to do.

There are many different academic and commercial projects for verifying smart contracts by developing their own smart contract programming languages, or by providing methods for verifying existing smart contract programming languages using proof assistants. One such project is Firmo which developed FirmoLang, a non-Turing complete domain specific language that has been formally verified using Coq [7]. A company named Secbit formalizes smart contracts for semi-automated verification and full formal verification on their platform [20]. Another project called Scilla developed an intermediate level smart contract language that embeds into the Coq proof assistant [11]. Lastly, a company named Certik is a formal verification platform that mathematically verifies smart contracts, although the details on how they achieve this are not specified to the

public yet [1].

6 Conclusion

Multiple projects have contributed to the use of proof assistants for DCP verification, especially recently with regard to blockchain consensus. Additionally, smart contracts are another use in the blockchain space where proof assistants are directly applicable. As the prevalence of distributed systems increases in our daily lives, proof assistants will become progressively more necessary to provide correct-by-construction proofs of the intended properties of these protocols.

References

[1] Certik. Formal verification platform. https://certik.org/, 2017.

[2] Bernadette Charron-Bost and Stephan Merz. Formal verification of a consensus algorithm in the heard-of model. *Int. J. Software and Informatics*, 3:273–303, 2009.

[3] Cezara DrăźGoi, Thomas A. Henzinger, Helmut Veith, Josef Widder, and Damien Zufferey. A logic-based framework for verifying consensus algorithms. In *Proceedings of the 15th International Conference on Verification, Model Checking, and Abstract Interpretation - Volume 8318*, VMCAI 2014, pages 161–181, Berlin, Heidelberg, 2014. Springer-Verlag.

[4] Burak et al. Ekici. Smtcoq: A plug-in for integrating smt solvers into coq. *Lecture Notes in Computer Science*, 10427, 2017.

[5] C. Mirto et al. Probabilistic formal methods applied to blockchain's consensus protocol. *BCRB '18 DSN Workshop on Byzantine Consensus and Resilient Blockchains*, 2018.

[6] K. Palmskog et al. Verification of casper in the coq proof assistant. *Project Report, November 15, 2018, Runtime Verification, Inc.*, 2018.

[7] FIRMO. Firmo: Secure execution of financial contracts. 2018.

[8] L. M. Goodman. Tezos: A self-amending crypto-ledger position paper. 2014.

[9] Hedera Hashgraph. Hedera hashgraph platform. https://www.hedera.com/platform, 2018.

[10] 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.

[11] A. Kumar I. Sergey and A. Hobor. Sscilla: a smart contract intermediate-level language. *CoRR*, abs/1801.00687, 2018.

[12] Marta Kwiatkowska, Gethin Norman, and Roberto Segala. Automated verification of a randomized distributed consensus protocol using cadence smv and prism? In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Computer Aided Verification*, pages 194–206, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[13] Leslie Lamport. Proving the correctness of multiprocess programs. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, SE-3(2), 1977.

[14] Leslie Lamport. Specifying concurrent systems with tla+. *Calculational System Design*, pages 183–247, April 1999.

[15] Oded Padon, Kenneth L. McMillan, Aurojit Panda, Mooly Sagiv, and Sharon Shoham. Ivy: Safety verification by interactive generalization. *SIGPLAN Not.*, 51(6):614–630, June 2016.

[16] R. Pass and E. Shi. Rethinking large-scale consensus. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 115–129, Aug 2017.

[17] George Pîrlea and Ilya Sergey. Mechanising blockchain consensus. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2018, pages 78–90, New York, NY, USA, 2018. ACM.

[18] Vincent et al. Rahli. Formal specification, verification, and implementation of fault-tolerant systems using eventml. *Electronic Communications of the EASST*, 72, 2015.

[19] Team Rocket. Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies. 2018.

[20] SECBIT. Formalizing a secure bit world starting from formalizing smart contracts. https://secbit.io/, 2018.

[21] Ilya Sergey, James R. Wilcox, and Zachary Tatlock. Programming and proving with distributed protocols. *Proc. ACM Program. Lang.*, 2(POPL):28:1–28:30, December 2017.

[22] Don et al. Tapscott. *The Blockchain Revolution*. 2016.

[23] Tatsuhiro Tsuchiya and André Schiper. Verification of consensus algorithms using satisfiability solving. *Distributed Computing*, 23(5):341–358, Apr 2011.

[24] James R. Wilcox, Doug Woos, Pavel Panchekha, Zachary Tatlock, Xi Wang, Michael D. Ernst, and Thomas Anderson. Verdi: A framework for implementing and formally verifying distributed systems. *SIGPLAN Not.*, 50(6):357–368, June 2015.