


Towards Formal Verification of DAG-Based Blockchain Consensus Protocols

Nathalie Bertrand ✉ 

Univ Rennes, Inria, CNRS, IRISA

Pranav Ghorpade ✉ 

The University of Sydney

Sasha Rubin ✉ 

The University of Sydney

Bernhard Scholz ✉ 

Fantom Research

Pavle Subotić ✉ 

Fantom Research

Abstract

There is a trend in blockchains to switch to DAG-based consensus protocols to decrease their energy footprint and improve security. A DAG-based consensus protocol orders transactions for delivering blocks, and relies on built-in fault tolerance communications via Byzantine Atomic Broadcasts. The ubiquity and strategic importance of blockchains call for formal proof of their correctness. We formalize the DAG-based consensus protocol called DAG-Rider in TLA+ and prove its safety properties with the TLA+ proof system. The formalization requires a refinement approach for modelling the consensus. In an abstracted model, we first show the safety of DAG-based consensus on leaders and then further refine the specification to encompass all messages for all processes. The specification consists of 683 lines and the proof system verifies 1922 obligations in about 5 minutes.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Proof theory; Theory of computation → Distributed algorithms

Keywords and phrases Formal verification, Consensus, Blockchain, Theorem Proving, TLA+

Consensus and DAG-based protocols. Consensus is a fundamental problem in distributed computing. It aims to coordinate processes so that they agree on some value(s). Consensus algorithms have recently become an important topic in “proof-of-stake” blockchains that collaboratively build an order for submitted transactions. Of particular interest are consensus algorithms that assume little about the environment, namely, asynchronous communications with malicious processes, namely Byzantine Fault Tolerant (BFT) [15].

Early blockchain consensus protocols assume degrees of synchrony in the environment to ensure safety and liveness [16, 2, 11, 8]. Recently, a family of probabilistic asynchronous consensus protocols have been introduced that are based on Directed Acyclic Graphs (DAG-based protocols) [10, 1, 6]. These protocols report high performance while guaranteeing BFT, utilize processes fairly, and exhibit low communication complexity. Several leading blockchains thus have adopted DAG-based protocols as their main consensus mechanism [4, 5, 9].

DAG-Rider [10] is such a DAG-based protocol and has two main components: (1) a communication layer and (2) an offline ordering layer. The communication layer asynchronously exchanges messages between processes in rounds using *reliable broadcast*. Messages contain transaction proposals and metadata forming a DAG for each node. For a process, the DAG provides a local view of the order of blocks with respect to happened-before relation [12]. Due to the asynchronous nature of the network, processes do not necessarily have the same local DAGs at any point in time. However they are guaranteed to have same DAGs eventually. The ordering layer selects anchor points, guaranteeing consistent selection across all the processes. This allows the DAGs to be locally totally ordered while guaranteeing that all the

47 processes agree on the same total order of messages.

48 **Formal verification of DAG-based consensus.** Blockchains provide mission-critical
49 financial services and hence require rigour to show correctness. The verification challenges arise
50 from the large number of possible interleaving in an asynchronous environment, the behaviours
51 of Byzantine processes, and perhaps even more importantly the fact that correctness should
52 hold for any number of participating processes.

53 We report here on our TLA+ [13] specification and proof –both publicly available at [7]–
54 for a DAG-based consensus protocol using the TLA+ Proof System (TLA-PS) [3].

55 Procedural code is commonly modeled in TLA+ by a discrete transition system whose
56 traces correspond to possible executions of the code. The naïve translation from the pseudo-
57 code (by setting every variable from the protocol, including a variable for each process’s
58 current line number, to be a variable in the specification) into a TLA+ specification is not
59 viable. While direct, this model is very fine-grained and renders the proofs extremely tedious.

60 To obtain a more succinct and tractable model, we employ several abstraction techniques:
61 they remove unnecessary details and produce a specification that is amenable to proofs. First,
62 we employ a *procedural abstraction* that ignores all states that are internal to a procedure and
63 only represents the input/output behaviour of each procedure in the DAG-Rider protocol.
64 For instance, in the *wave_ready* procedure of [10], the relevant variables are *decidedWave*,
65 *deliveredVertices*, *leadersStack*, but not the loop variable w' or the auxiliary variable v' .
66 Second, because we focus on safety properties, we remove component features that are only
67 required for liveness and have no impact on the safety proof. For instance, random coin tosses
68 can be replaced with deterministic ones. Third, we use memoization to efficiently compute
69 the values taken by recursive functions, by introducing a fresh state variable that stores the
70 needed information to evaluate recursive functions in a single step. Finally, we separate the
71 concerns and break the safety property into two, namely (1) consistent communication and
72 (2) consistent leader election. For (1) we model the DAG-construction and show that the
73 causal histories agree for a same vertex in the DAG of two different processes¹. For (2), we
74 model the consensus protocol and prove that the same leaders are elected and in the same
75 order. To obtain a complete yet simple model of the consensus protocol, we observe that it
76 only needs reachability information associated with wave leader vertices to commit leaders
77 and, therefore, abstract the content of DAG into the so-called *leaderReachability* record. We
78 combine consensus protocol specifications in DAG construction specifications to obtain one
79 of the DAG-Rider protocols. This abstraction is not only interesting for DAG-Rider but
80 could be helpful to generalize to other DAG-based protocols.

81 Given our faithful specification of DAG-Rider in TLA+, we prove its expected safety
82 properties by identifying invariants and proving them within TLA-PS. When using TLA-PS
83 and similar proof systems, the most challenging task is to come up with relevant inductive
84 invariants (that hold initially and are preserved when taking transitions), see for instance [14].
85 For DAG-Rider, to prove the consistency of communication during the DAG construction
86 we identified 6 new invariants, and to prove the consistency of leader election we identified
87 10 new invariants. We prove each one of the invariants hierarchically by induction.

88 Table 1 provides some metrics on our experiments, showing quite reasonable performances
89 in terms of verification time. Most importantly, due to the modularity of our specification, we
90 argue the effort to adapt proofs is minimal when making small changes to the specification.

91 **Conclusion.** Our work on DAG-Rider is an important and promising step towards a
92 general library for specifying and verifying DAG-based consensus protocols. Beyond the

¹ The non equivocation of blocks is guaranteed by reliable broadcast abstraction

■ **Table 1** Summary of experiments. An obligation is a condition that TLA-PS checks. The time to check is on a 2.10 GHz CPU with 8 GB of memory, running Windows 11 and TLA-PS v1.4.5.

Metric	DAG-Constr. Spec.	Consensus Spec.	DAG-Rider Spec.
Size of spec. (# loc)	460	250	710
Size of proof (# loc)	521	782	1303
Max level of proof tree nodes	10	9	10
Max degree of proof tree nodes	7	7	7
# obligations in TLA-PS	722	1205	1927
Time to check by TLA-PS (s)	224	87	311

93 specification of DAG-Rider, our specification reveals interesting insights into developing a
94 modular and efficient TLA+ specification that is amenable to proofs in TLA-PS.

95 ——— References ———

- 96 1 Leemon Baird and Atul Luykx. The hashgraph protocol: Efficient asynchronous BFT for
97 high-throughput distributed ledgers. In *Proceedings of COINS 2020*, pages 1–7. IEEE, 2020.
98 doi:10.1109/COINS49042.2020.9191430.
- 99 2 Vitalik Buterin. Ethereum white paper: A next generation smart contract & decentralized
100 application platform, 2013. URL: <https://github.com/ethereum/wiki/wiki/White-Paper>.
- 101 3 Denis Cousineau, Damien Doligez, Leslie Lamport, Stephan Merz, Daniel Ricketts, and
102 Hernán Vanzetto. TLA + proofs. In *Proceedings of FM 2012*, volume 7436 of *Lecture Notes*
103 *in Computer Science*, pages 147–154. Springer, 2012. doi:10.1007/978-3-642-32759-9_14.
- 104 4 Aptos Foundation. Understanding Aptos: A comprehensive overview, 2024. URL: <https://messari.io/report/understanding-aptos-a-comprehensive-overview>.
- 105 5 Fantom Foundation. Lachesis aBFT, 2024. URL: <https://docs.fantom.foundation/technology/lachesis-abft>.
- 106 6 Adam Gagol, Damian Lesniak, Damian Straszak, and Michal Swietek. Aleph: Efficient atomic
107 broadcast in asynchronous networks with byzantine nodes. In *Proceedings of AFT 2019*, pages
108 214–228. ACM, 2019. doi:10.1145/3318041.3355467.
- 109 7 Pranav Ghorpade. TLA+ specification and Proofs for DAG-Rider. <https://github.com/pranavg5526/DAG-Rider>, 2024. [Online; accessed 1-Feb-2024].
- 110 8 Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand:
111 Scaling Byzantine agreements for cryptocurrencies. In *Proceedings of SOSP 2017*, pages 51–68.
112 ACM, 2017. doi:10.1145/3132747.3132757.
- 113 9 Hedra. Streamlining consensus, 2024. URL: <https://hedera.com/blog/streamlining-consensus-throughput-and-lower-latency-with-about-half-the-events>.
- 114 10 Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. All you need
115 is DAG. In *Proceedings of PODC 2021*, pages 165–175. ACM, 2021. doi:10.1145/3465084.
116 3467905.
- 117 11 Jae Kwon. Tendermint: Consensus without mining. <https://tendermint.com/docs/tendermint.pdf>, 2014.
- 118 12 Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun.*
119 *ACM*, 21(7):558–565, jul 1978. doi:10.1145/359545.359563.
- 120 13 Leslie Lamport. *Specifying Systems, The TLA+ Language and Tools for Hardware and Software*
121 *Engineers*. Addison-Wesley, 2002. URL: <http://research.microsoft.com/users/lamport/tla/book.html>.
- 122 14 Leslie Lamport. Teaching concurrency, 2009. URL: <https://lamport.azurewebsites.net/pubs/teaching-concurrency.pdf>.
- 123 15 Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM*
124 *Trans. Program. Lang. Syst.*, 4(3):382–401, jul 1982. doi:10.1145/357172.357176.
- 125 16 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.