(Auto)Complete this Proof: Decentralized Proof Generation via Smart Contracts

Jin Xing Lim[1], Barnabé Monnot[2], Georgios Piliouras[1], and Shaowei Lin[3]

[1] Singapore University of Design and Technology, {jinxing_lim@mymail., georgios@}sutd.edu.sg
[2] Ethereum Foundation, barnabe.monnot@ethereum.org
[3] shaowei@gmail.com

Automated theorem proving has come a long way in the last few decades. Multiple *automated theorem provers* (ATPs) have been developed, such as E [15] and Vampire [13] in the case of first-order logic (FOL). For higher-order logic, automated tools exist for *interactive theorem provers* (ITPs), such as Sledgehammer [11] (for Isabelle [18]), TacticToe [9] (for HOL4 [16]), CoqHammer [7] and Tactician [4] (both for Coq [3]). The effectiveness, however, of these automated tools is still not as desirable as one would hope. Paulson and Blanchette show in their experiments that Sledgehammer only managed to automate 52% of the proofs from seven selected Isabelle theories [12]. On the brighter side, combinations of different automated tools yield better results than the tools individually. In the experiments run by the developers of Proverbot9001 [14], the combination of CoqHammer and their tool is 32% more effective in automated proving than the tool itself. Furthermore, they concluded that CoqGym [20] and their approaches are complementary as both can complete proofs which the other cannot. Thus, this raises our driving challenge: *to design a system architecture that allows for the effective collaboration between multiple intelligent agents (humans and AI systems) towards producing formally correct proofs and programs.*

Our approach. We propose a blockchain-based protocol for proof and program synthesis through some ITP such as Coq. This approach allows both for the design of open problems via intents which have not been formally completed as well as for the storage of partial proofs. Using token-based mechanisms, we incentivize continued participation and completion of partial proofs. Furthermore, we encourage the development and application of generic proof tactics which are useful in completing numerous diverse theorems/programs.

Our architecture instantiates a top-down approach to proving theorems: provers publish partial proofs, formally correct up to specific unproven lemmas. To coordinate and distribute claims and partial proofs, the architecture features a programmable blockchain. Claims of first authorship are verified with time-stamped records on the chain. Smart contracts define complex incentive schemes to reward multiple collaborators, e.g., depending on how peers evaluate their respective contributions. Blockchain data is highly structured and available, ideal for AI agents to iterate over and attempt to solve unproven claims, possibly in collaboration with one another. Other than storing multiple copies of mathematical proofs and formally verified programs at different nodes, the decentralization critically allows for the emergence and persistence of *common knowledge* [8], which mirrors the process of academic publishing and public presentation of results.

Sketch of the system architecture

We assume the availability of a public, programmable blockchain, where smart contracts are deployed. We call *prover* an account (or address) on the blockchain participating in the protocol. A human or AI participant may operate several prover accounts, although for simplicity we consider the case where one entity corresponds to one prover only.

Data Layer

Provers add time-stamped *records* of *contributions* on the blockchain. The record is a simple data structure registered in the blockchain's state. Minimally, the record contains (a) the prover's address and (b) a reference to the contribution. In our protocol, the contribution is a Coq file containing additional metadata. Metadata attributes include the contribution type, among three distinct possibilities: (1) a conjecture (proposition/type with empty proof), (2) partial proof to some earlier conjecture (leaving sub-conjectures for other agents to complete), or (3) completed mathematical objects such as new definitions, propositions and tactics. The metadata also includes references to previous relevant contributions (e.g., "imports"). While records are stored on the blockchain, contributions are stored on a decentralized file storage infrastructure, such as IPFS [2], with the hash of the contents of the file used as the contribution's reference in its corresponding record.

Client Layer

Provers access records and contributions via the client layer. A client is an interface responsible for downloading records and contributions in a structured manner, e.g., by following the directed acyclic graph (DAG) of contributions built by imports declared in the metadata of a single contribution. While downloading contributions, clients perform validity checks that cannot be handled on the data layer, e.g., check that the Coq code is syntactically correct. Invalid contributions may either be flagged by the client or disregarded entirely. A client interface may be built, e.g., as a plug-in to popular Coq code editors, allowing provers to add structured metadata as well as publishing records and contributions to the blockchain. Additionally, the interface would present data derived from the incentive layer, introduced in the next section.

Incentive Layer

Value representation is handled natively by public, programmable blockchains, allowing the protocol to maintain for each prover a *balance* denominated in *tokens*. Our protocol uses incentives in a flexible manner: any prover can deploy some incentive mechanism as a smart contract, as long as the mechanism makes reference to contribution records. For instance, an organization can deploy a smart contract containing a reward for the proof of a theorem. The smart contract allows one prover only, e.g., a registered prover ("judge") belonging to the organization, to declare a previously published record as the winner of the prize, after which the balance of the winner is incremented with the prize tokens. The decision to offer the prize may be up to a vote by a quorum of provers in the organization, or provers who were certified by another set of provers as having made significant contributions in the past (such mechanisms "seed" an initial set who is responsible for co-opting new members, growing over time). The quorum could also split the reward via some allocation rule to incentivize the contribution of partial proofs.

To score contributions, different approaches could explore distinct trade-offs between human input and contextual scoring. For example, one approach could deem a contribution valuable whenever some index of its centrality within the contribution DAG is high (e.g., PageRank [5]). Another approach is to rank a contribution as more valuable if there are more provers declare that it is. For instance, Token-Curated Registries (TCRs) [10, 1] incentivize participants to vote and maintain a list according to some agreed upon criteria.

Comparison with related works

There are several prior projects such Qeditas [19], Mathcoin [17] and Proofgold [6] that discussed on how we could use blockchain mechanism to tie sources of formalized mathematics together in a decentralized way. To encourage participation, they discuss ideas such as rewarding creators with ownership rights and/or some form of currency. What distinguishes our idea from them is the proposal of having smart contracts to introduce incentive mechanisms. With appropriate incentive mechanism in place, we could gamify the process of formalizing mathematics and award contributors with virtual tokens for partial or completed results they have done. A particular importance is the fact that we can design complex incentive mechanisms that can incentivise multi-agents collaboration. The balance of these tokens can either be exchanged for real-life currency (e.g. US dollars) or be used to measure each mathematician's or computer scientist's contribution to formalized mathematics.

During AITP'21, we would be presenting a proof of concept to see how different formalized proofs of a sorting specification written in the Coq proof assistant can be generated from human-AI collaboration via a blockchain mechanism. We would like to gather feedback about our approach and discuss other techniques that may suit this framework.

References

[1] Aditya Asgaonkar and Bhaskar Krishnamachari. Token curated registries-a game theoretic approach. *arXiv preprint arXiv:1809.01756*, 2018.

[2] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

[3] Yves Bertot. A short presentation of coq. In *International Conference on Theorem Proving in Higher Order Logics*, pages 12–16. Springer, 2008.

[4] Lasse Blaauwbroek, Josef Urban, and Herman Geuvers. The tactician. In *International Conference on Intelligent Computer Mathematics*, pages 271–277. Springer, 2020.

[5] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. 1998.

[6] Chad E. Brown. A theory supporting higher-order abstract syntax. Technical report, Tech. rep., Czech Technical University in Prague (Aug 2020), 2020.

[7] Łukasz Czajka and Cezary Kaliszyk. Hammer for coq: Automation for dependent type theory. *Journal of automated reasoning*, 61(1-4):423–453, 2018.

[8] Ronald Fagin, Yoram Moses, Joseph Y Halpern, and Moshe Y Vardi. *Reasoning about knowledge*. MIT press, 2003.

[9] Thibault Gauthier, Cezary Kaliszyk, and Josef Urban. Learning to reason with hol4 tactics. *arXiv preprint arXiv:1804.00595*, 2018.

[10] Mike Goldin. Token-curated registries 1.0. `https://docs.google.com/document/d/1BWWC__-Kmso9b7yCI_R7ysoGFIT9D_sfjH3axQsmB6E/edit`, 2017. Accessed: 2021-01-20.

[11] Jia Meng, Claire Quigley, and Lawrence C Paulson. Automation for interactive proof: First prototype. *Information and computation*, 204(10):1575–1596, 2006.

[12] Lawrence Paulson and Jasmin Blanchette. Three years of experience with sledgehammer, a practical link between automatic and interactive theorem provers. 02 2015.

[13] Alexandre Riazanov and Andrei Voronkov. The design and implementation of vampire. *AI communications*, 15(2, 3):91–110, 2002.

[14] Alex Sanchez-Stern, Yousef Alhessi, Lawrence Saul, and Sorin Lerner. Generating correctness proofs with neural networks. In *Proceedings of the 4th ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 1–10, 2020.

[15] Stephan Schulz. E–a brainiac theorem prover. *Ai Communications*, 15(2, 3):111–126, 2002.

[16] Konrad Slind and Michael Norrish. A brief overview of hol4. In *International Conference on Theorem Proving in Higher Order Logics*, pages 28–32. Springer, 2008.

[17] Borching Su. Mathcoin: A blockchain proposal that helps verify mathematical theorems in public. *IACR Cryptol. ePrint Arch.*, 2018:271, 2018.

[18] Makarius Wenzel, Lawrence C Paulson, and Tobias Nipkow. The isabelle framework. In *International Conference on Theorem Proving in Higher Order Logics*, pages 33–38. Springer, 2008.

[19] B White. Qeditas: A formal library as a bitcoin spin-off (2016). *URL http://qeditas.org/docs/qeditas.pdf*, 2016.

[20] Kaiyu Yang and Jia Deng. Learning to prove theorems via interacting with proof assistants. *arXiv preprint arXiv:1905.09381*, 2019.