Why Bitcoin Needs a Specification That Humans Can Read

Josh · Secure Sovereign

Published July 7, 2026 · Updated September 20, 2026

Verification methodology is not just a technical question but a governance question that will shape Bitcoin's resilience for decades.

Contents

The Problem Bitcoin Has Been Avoiding

Bitcoin has been extraordinarily successful without a formal specification. But success does not make the absence of one sustainable.

The consensus rules that secure a multi-trillion-dollar network exist only as behavior encoded in Bitcoin Core's C++ codebase. **"The code is the spec" is not a design principle but an admission that no one has written the spec.** Every alternative implementation, every audit, every attempt to verify correctness must reverse-engineer undocumented behavior from a codebase that was never designed to be read as a specification.

This has worked so far because Bitcoin's growth in economic importance has outpaced the scrutiny that importance attracts. That is changing. When a court needs to understand whether a transaction was valid under the consensus rules at a given block height, there is no document to hand them. When a new implementation team wants to be sure they are building the same rules as Core, there is no specification to target. When an auditor wants to verify that a proposed soft fork does not break existing guarantees, there is no formal description of what those guarantees are.

The absence of a specification is not a minor gap. **It is the foundational vulnerability that every other correctness problem in Bitcoin builds on.**

The missing spec also keeps technical questions in the social layer indefinitely, and the noise degrades the signal from genuine threat detection. For why that noise-floor problem is an argument for a formal spec without asking Bitcoin to soften its adversarial register, see *The Adversarial Default, §IV*.

Three Responses to the Same Problem

Several serious efforts are underway to address this, and they deserve to be understood on their own terms.

The first is the human-readable mathematical specification approach, which is what Bitcoin Commons has built with the Orange Paper. The spec is written in standard mathematical notation, readable by any mathematician without knowledge of any programming language or proof assistant. The implementation is written separately in Rust and checked against it continuously. Neither artifact depends on the other to be understood. For how formal caps fit into a spec-first blockspace floor, see *The Achievable Floor*.

The second uses Lean 4, a proof assistant, to build machine-checked guarantees. btc-verified, an independent project by Keagan McClelland, takes this approach: verified proof leaves on Bitcoin's byte layer, covering codecs, serialization, merkle, transaction ids, and chain linkage. The proofs verify mathematical properties and stop there. No running node, no generated code. A more ambitious variant would use proofs to generate the implementation itself through a compilation pipeline, though no project ships this at Bitcoin node scale today.

The third is the declarative DSL model taken by Hornet Node, developed by Toby Sharp. Hornet Node is a declarative C++ client written from scratch with no copied Core code, capable of syncing mainnet to tip in a few hours on a single thread. Its companion DSL is designed to eventually generate and validate implementation code, though that capability is still in development.

All three are good-faith responses to a real problem. The question is which foundation is best for Bitcoin over the timescales that actually matter.

Verification Is a Governance Problem

This is the argument that changes the conclusion, and it is not being made anywhere else in this debate.

Bitcoin Core's governance problem is well understood in this community. Merge authority over the only implementation that the vast majority of economic nodes run is concentrated in a very small group of people. The problem is not that those people are bad engineers. **Concentrating consequential authority over Bitcoin in any small group, regardless of their competence or intentions, creates a single point of capture.** For the no-spec moat and why alternatives stay

tethered to Core, see *Who Controls Bitcoin, §VI*. For common defenses of the implementation monopoly that treat the code as sufficient specification, see *Bitcoin Core: The Biggest Fallacies* (Fallacy 6).

The choice of verification methodology is the same problem at a different layer.

A verification approach that requires rare, specialized knowledge concentrates review authority in whoever holds that knowledge. If Bitcoin's correctness story depends on Lean proofs, its security model depends on a small population of people who can read and audit those proofs well enough to catch a subtle error in a consensus-critical theorem. That population is small, concentrated in academic communities, and not distributed across the independent institutions Bitcoin's long-term security requires. A custom DSL creates the same problem: the reviewer population is bounded by whoever understands the generator.

Standard mathematical notation is different in kind, not just in degree. It is not a tool requiring training in a specific software ecosystem but a communication medium that has been the universal language of precise formal reasoning for five hundred years, predating every programming language, every proof assistant, and every computer. Any mathematician with Bitcoin domain knowledge can read the Orange Paper and evaluate whether the stated rules are correct. That population is orders of magnitude larger, and it is stable across generations in a way that expertise in any specific tool cannot be.

The choice of verification methodology is not just about which technique catches more bugs today. It is about which approach maximizes the number of independent people who can meaningfully evaluate Bitcoin's rules over decades. Every verification layer that requires specialized tool knowledge narrows that population and concentrates authority. The Orange Paper widens it permanently.

Bitcoin Commons exists to dissolve implementation concentration. The same principle governs how we think about verification.

Defense in Depth, Not a Silver Bullet

Framing this as a choice between human readability and formal rigor misses the point. The right answer is a stack.

Bitcoin Commons's verification approach has four layers. The Orange Paper is the normative reference: human-readable, tool-agnostic, auditable by any mathematician, targetable by any implementation team. The BLVM Specification Lock uses `Z3` to check on every merge that the Rust implementation still satisfies the mathematical contracts derived from the spec. Differential testing runs the implementation against Bitcoin Core across the full mainnet history. Those full-chain runs are operator-driven and resource-heavy. That is not a claim every CI job has zero-

divergence proof to tip. Machine-checked proof leaves, following btc-verified's methodology, strengthen the byte layer where formal methods are most tractable today and will expand upward as tools mature.

The Specification Lock deserves particular attention because it solves the code generation problem without introducing one of its own.

In a code generation system, the generator is the authoritative artifact. The implementation is derived from it. When something is wrong in the generated code, the fix travels through the generator first. A change to the generator can produce unexpected effects anywhere in its output. Every DSL and proof extraction pipeline has encountered edge cases the generator does not handle. The resolution is always the same: fix the generator, regenerate, verify again. The debugging surface is not the code but the thing that made the code.

The Specification Lock inverts this. **The implementation is the authoritative artifact.** The Specification Lock reads the implementation against the specification rather than producing the implementation from it. When a check fails, the developer fixes whichever side is wrong, directly, without touching a generator. Nothing is regenerated. The Orange Paper can be read without the implementation. The implementation can be audited without the Orange Paper. The Specification Lock is the enforced relationship between them, not the thing that produces either one.

This is how every durable technical standard works. TCP/IP, TLS, POSIX, and the C standard are human-readable specifications implemented independently by teams who never needed to run the specification as code.

The Timescale That Actually Matters

Bitcoin is infrastructure that must survive institutional and generational turnover. This changes which trade-offs matter.

In the nearer 50 to 100 year window, the question is whether a verification approach can be maintained across the institutional change Bitcoin governance will require. Lean 4 is fifteen years old and actively evolving. Proofs written today require active maintenance as the toolchain develops. A DSL compiler is additional software with its own maintenance burden. Mathematical notation requires no such maintenance. It does not depend on any software remaining operational or any organization remaining funded.

Over longer horizons, the durability gap widens. Legislators, regulators, central banks, and courts will increasingly need to understand what Bitcoin's consensus rules actually are. They will not hire Lean experts. They will hire mathematicians and lawyers who can read a specification written in standard notation. The institutional audiences that will govern Bitcoin's relationship with human society for decades are already reachable through the Orange Paper in a way they are not through any proof assistant.

This is not an argument against formal methods. Machine-checked proofs provide guarantees that empirical testing cannot. The Orange Paper provides the human-legible normative foundation that proofs cannot replace. Differential testing grounds the whole stack in real chain history. Each layer does what the others cannot. The argument is for being deliberate about which layer carries which burden.

The Path We Are Taking

Bitcoin Commons is not claiming to have solved this problem. The Orange Paper is a first formal specification of Bitcoin consensus, not the last word on it. Specification Lock coverage will expand. Formal proof leaves on the byte layer are being incorporated. The differential testing program will continue to grow. Cryptographic merge authorization is designed and stays off until security review, key management, and community validation.

What we are claiming is that the foundation matters, and that getting it right is the prerequisite for everything else. Implementation diversity on the same chain means multiple independent implementations, built by different teams in different languages, each developed against the same specification, each auditable by the same population of mathematicians, none of them the sole source of the rules. The rules stay legible to the engineers, regulators, and institutions that will need them, without requiring specialist intermediaries.

That is what implementation diversity actually requires. **Not a better single implementation, but a specification that outlives every implementation built against it.** Building that specification is what Bitcoin Commons is doing, and we think it is the most important work in Bitcoin right now.

Sources

- Orange Paper / Bitcoin Commons consensus spec
- Bitcoin Core repository

Related: Who Controls Bitcoin, The Vertical Layer Problem, Making Core Irrelevant, The Achievable Floor, Bitcoin Governance: Argument Map, Permanent Data Channel Closure.

SECURESOVEREIGN
Markdown is the canonical source: https://secsov.com/articles/why-bitcoin-needs-a-specification/index.md
Formatted snapshot of https://secsov.com/articles/why-bitcoin-needs-a-specification
Josh · Secure Sovereign