Failure-Aware Reachability Certificates for QA: Unified Proof and Projection Artifacts

Will Dale

January 2026

Abstract

We introduce a *proof certificate* schema for Quantum Arithmetic (QA) that treats both successful derivations and certified obstructions as first-class mathematical objects. Unlike standard proof traces which only show success, our certificates always emit a reviewable obstruction witness when search fails (e.g., depth exhaustion, invariant violation, component disconnection). We further extend the same certificate machinery to projection-dependent physics experiments, where classical laws emerge only under specific observers. All claims are packaged as deterministic JSON artifacts.

1 Motivation

Two failures repeatedly block reproducible claims in symbolic / geometric proof search: (i) successes are traceable, but failures are not, and (ii) "physics-like" behavior in QA is often asserted without a falsifiable projection contract. We address both by standardizing *certificates*.

2 Certificate Objects

Define a certificate as a tuple:

$$\mathcal{C} = (\texttt{generator\_set}, \texttt{contracts}, \texttt{witness\_type}, \texttt{witness})$$

where all reachability statements are relative to a generator universe.

2.1 Generators and Namespaces

QA substrate generators use the canonical set $\{\sigma, \lambda, \mu, \nu\}$ (and optional inverses). External engines are namespaced:

$$\texttt{AG:rule\_id}, \quad \texttt{PHYS:probe}.$$

2.2 Invariants and Non-Reduction

Certificates optionally track a list of invariants that must be preserved. QA's Non-Reduction Axiom is represented as a contract flag; violations are certified as `NON_REDUCTION_VIOLATION`.

2.3 Failure Taxonomy

We treat failure modes as a finite algebra of obstructions (not bugs), including: `DEPTH_EXHAUSTED`, `GENERATOR_INSUFFICIENT`, `SCC_UNREACHABLE`, etc.

3 Results: Integration with AlphaGeometry

We demonstrate the certificate framework by integrating it with *AlphaGeometry*, a state-of-the-art automated geometry prover. Our goal is not to modify AlphaGeometry's proof search, but to *externalize its outcomes*—both successes and failures—into a uniform, first-class certificate object.

3.1 Adapter Overview

AlphaGeometry produces a `SearchResult` object containing: (i) a Boolean solved flag, (ii) an optional proof trace, and (iii) search statistics (depth, expansions, policy). We implement a deterministic adapter that maps each `SearchResult` into a `ProofCertificate` without altering search behavior.

Key design choices are:

- **Namespaced generators.** Each AlphaGeometry rule is represented as a generator `AG:<rule_id>`, ensuring disjointness from QA-native generators $(\sigma, \lambda, \mu, \nu)$ and physics observers.
- **Exact, minimal state references.** AlphaGeometry states lack numeric QA coordinates; we therefore use stable hash-based identifiers to reference states without inventing geometry.
- **Conservative failure classification.** When a search terminates without proof, the adapter produces an obstruction certificate but does not claim strong properties (e.g., unreachable SCCs) unless explicitly justified by evidence.

The adapter produces either a *success certificate* with a witness path, or an *obstruction certificate* with explicit failure evidence, using the same schema in both cases.

3.2 Success Certificate: Parallel Transitivity

Our first example is a successful AlphaGeometry proof of the *parallel transitivity* theorem. The exported certificate (`parallel_transitivity_proof.cert.json`) contains:

- A single-step witness path using generator `AG:parallel_transitivity`.
- Deterministic source and destination state identifiers.
- An invariant contract with `non_reduction_enforced = false`, reflecting AlphaGeometry's independent algebraic semantics.

Formally, this certificate is a reachability witness:

$$s_0 \xrightarrow{\texttt{AG:parallel\_transitivity}} s_1,$$

where the generator set explicitly records the rule required for the proof. No additional invariants or observers are assumed.

This artifact demonstrates that trivial and non-trivial AlphaGeometry proofs can be embedded into the certificate framework without loss of semantic fidelity.

3.3 Obstruction Certificate: Unsolvable Configuration

To demonstrate failure handling, we consider an intentionally unsolvable geometry configuration. AlphaGeometry terminates immediately, generating no successors and producing no proof trace.

The corresponding certificate (`unsolvable_obstruction.cert.json`) is an *obstruction certificate* with:

- `witness_type = obstruction`,
- `fail_type = depth_exhausted`,
- Explicit evidence: zero successors generated at depth zero.

Crucially, the certificate does *not* claim that the target state is globally unreachable under all generators; it certifies only what is known: that under the given generator set and search policy, no progress was possible.

This conservative classification avoids overclaiming while still producing a reusable, inspectable artifact.

3.4 Unified Interpretation

These two results illustrate the central claim of the framework:

Success and failure are objects of the same type, differing only in witness content.

Both certificates: (i) use the same schema, (ii) explicitly record their generator assumptions, and (iii) are reproducible and machine-checkable.

In §4, we show that the same certificate structure extends beyond theorem proving, capturing physical laws and their failures as projection-dependent reachability phenomena.

4 Physics as Projection

We add an explicit `ProjectionContract` recording: observer id, time projection, topology/symmetry preservation, and projected observables. A classical law is then stated as: *the law holds under observer O.*

Artifacts.

- `reflection_GeometryAngleObserver.success.cert.json`
- `reflection_NullObserver.obstruction.cert.json`

5 Discussion

Certificates shift evaluation from narrative claims to artifact inspection: success witnesses are checkable, and failures become reviewable mathematical statements. The same mechanism unifies formal proof, reachability, and projection-dependent physical laws.

6 Conclusion

QA can support a certificate-first scientific workflow: every claim ships as deterministic JSON, with a canonical schema and adapters for multiple domains.