The QA Decision Certificate Spine: Machine-Checkable Witnesses for Sequential Decision Making

Signal Experiments Research Group

January 21, 2026

Abstract

We present the *QA Decision Certificate Spine*, a unified framework that provides machine-checkable certificates for all major layers of sequential decision making: probabilistic inference, optimal planning, online search, exploration, state estimation, reinforcement learning, and imitation learning. Unlike existing approaches where failures are opaque training artifacts, our framework treats *failure as a first-class mathematical object*—every unsuccessful computation produces a constructive obstruction witness explaining why success was impossible. We demonstrate complete coverage of the MIT *Algorithms for Decision Making* curriculum (Chapters 3–13) with 295 validated test cases, cross-certificate coherence checking, and recompute hooks enabling full auditability. The framework uses exact rational arithmetic throughout, eliminating floating-point uncertainty from certificate validation.

1 Introduction

Sequential decision making under uncertainty is fundamental to robotics, autonomous systems, and artificial intelligence. Despite decades of progress, current systems lack a crucial property: *when they fail, they cannot explain why.*

Consider a reinforcement learning agent that fails to converge. Current frameworks report "training did not converge" or "reward threshold not reached." But *why* did it fail? Was the target unreachable? Did exploration collapse prematurely? Was the reward non-identifiable?

We introduce the **QA Decision Certificate Spine**, a framework where every computation—successful or not—produces a *machine-checkable certificate*. Successes come with verifiable witnesses; failures come with constructive obstruction evidence.

1.1 Contributions

1. **Unified Certificate Architecture**: Seven certificate types covering inference, planning, search, exploration, filtering, learning, and imitation—all sharing common validation semantics.

2. **Failure as First-Class Object**: Every failure mode produces obstruction evidence with the same mathematical status as success witnesses.

3. **Exact Arithmetic**: All certificates use rational arithmetic ($\mathbb{Q}$), eliminating floating-point validation ambiguity.

4. **Cross-Certificate Coherence**: Bundle validation ensures consistency across certificate types (e.g., RL success implies policy feasibility).

5. **Recompute Hooks**: Critical certificates include deterministic recomputation functions for full auditability.

2 Related Work

Formal Verification in ML Approaches like [1] verify neural network properties but do not address sequential decision making. Our work extends formal methods to the full decision pipeline.

Probabilistic Programming Systems like Stan and Pyro provide inference but lack certificate infrastructure for failures. Our `InferenceCertificate` fills this gap.

Safe Reinforcement Learning Constrained RL [2] optimizes under safety constraints but does not provide obstruction certificates when constraints cannot be satisfied.

Provably Efficient RL PAC-MDP guarantees [3] provide sample complexity bounds but not per-instance failure explanations.

3 The Certificate Spine Architecture

Definition 1 (QA Certificate)**.** *A* QA Certificate *is a tuple* (M, T, W, V) *where:*

- M *is a model specification (state space, actions, dynamics)*
- T *is a task specification (target, constraints, horizon)*

- W *is a witness (success proof or obstruction evidence)*
- $V : (M, T, W) \to \{\texttt{valid}, \texttt{invalid}\}$ *is a deterministic validator*

Definition 2 (Obstruction Evidence)**.** *An* obstruction *is a witness that proves task T cannot be achieved under model M. Unlike a mere failure flag, an obstruction is a constructive mathematical object.*

3.1 Certificate Types

Table 1 summarizes the seven certificate types and their correspondence to decision-making chapters.

Table 1: Certificate types and their QA-native features

Certificate	Chapter	QA Feature	Recompute
Inference	3–4	VE = graph reduction	✓
Policy	7	BFS = shortest path	–
MCTS	8	SCC pruning witness	–
Exploration	9	Regret = steps − BFS	–
Filter	9–11	Kalman/particle	✓
RL	12	Reward = distance delta	✓
Imitation	13	IRL = target inference	–

4 Inference Certificates (Chapters 3–4)

Definition 3 (InferenceCertificate)**.** *An inference certificate attests to probabilistic inference over a factor graph:*

$$\texttt{InferenceCert} = (G, Q, E, \mu, P)$$

where G is a factor graph, Q are query variables, E is evidence, $\mu : Q \to \mathbb{Q}$ is the computed marginal (exact rational), and P is a method proof.

4.1 Failure Modes

1. **TREEWIDTH_TOO_HIGH**: Exact inference intractable; obstruction includes treewidth lower bound.
2. **MESSAGE_DIVERGENCE**: Belief propagation did not converge; obstruction includes divergence trajectory.
3. **EVIDENCE_INCONSISTENT**: $P(\text{evidence}) = 0$; obstruction proves zero probability.

4.2 Recompute Hook

```
RecomputeVE(cert, factors):
    for v in elimination_order:
        factors = eliminate(v, factors)
    mu_prime = normalize(factors)
    return mu_prime == cert.mu
```

5 Policy Certificates (Chapter 7)

Definition 4 (PolicyCertificate)**.** *A policy certificate attests that policy π achieves target T from initial state s_0:*

$$\texttt{PolicyCert} = (\pi, s_0, T, d^*, P)$$

where d^* is the optimal distance (BFS path length) and P is the optimality proof.

QA-Native Insight In the QA framework, BFS optimal distance equals the shortest reachability path in the generator graph. This connects classical planning to algebraic reachability.

5.1 Failure Modes

1. **NO_PATH_EXISTS**: Target unreachable; obstruction includes reachability analysis showing target outside forward reachable set.
2. **HORIZON_EXCEEDED**: Path exists but exceeds horizon; obstruction includes shortest path length.
3. **OBSTRUCTION**: Path blocked by obstruction class; includes obstruction witness.

6 MCTS Certificates (Chapter 8)

Definition 5 (MCTSCertificate)**.** *An MCTS certificate attests to online planning via tree search:*

$$\texttt{MCTSCert} = (s_0, a^*, Q, \texttt{SCC}, P)$$

where a^* is the best action, Q are action values, SCC is the SCC pruning witness, and P is the method proof.

6.1 SCC Pruning Witness (QA Differentiator)

The key QA-native feature is *SCC pruning*: strongly connected component analysis of the state graph allows provable pruning of unreachable subtrees during rollout.

Definition 6 (SCC Pruning Witness)**.**

$$\texttt{SCCWitness} = (h, n_{pruned}, \{S_i\}_{unreachable})$$

where h is the SCC computation hash, n_{pruned} is nodes pruned, and $\{S_i\}$ are unreachable SCC identifiers.

Pruning Efficiency

$$\eta = 1 - \frac{n_{\text{QA}}}{n_{\text{vanilla}}}$$

measures the fraction of rollouts saved by SCC pruning.

7 Exploration Certificates (Chapter 9)

Definition 7 (ExplorationCertificate)**.** *An exploration certificate attests to exploration-exploitation performance:*

$$\texttt{ExploreCert} = (M, R, P)$$

where M is the exploration method, R is the regret witness, and P is the method proof.

7.1 Regret Witness (QA-Native)

Definition 8 (Regret Witness)**.**

$$\texttt{RegretWitness} = (n_{actual}, n_{optimal}, r_{cumulative}, \mathcal{O}(\cdot))$$

where $r_{cumulative} = n_{actual} - n_{optimal}$ and $\mathcal{O}(\cdot)$ is the regret bound.

QA-Native Insight Regret is concretely defined as (steps to target) − (BFS optimal steps). This grounds abstract regret in reachability geometry.

8 Filter Certificates (Chapters 9–11)

Definition 9 (FilterCertificate)**.** *A filter certificate attests to state estimation:*

$$\texttt{FilterCert} = (D, \hat{x}, \Sigma, P)$$

where D is the dynamical system, $\hat{x}$ is the state estimate, Σ is uncertainty (covariance or credible interval), and P is the method proof.

8.1 Failure Modes

1. **PARTICLE_DEGENERACY**: Effective sample size (ESS) below threshold; obstruction includes ESS trajectory.
2. **STATE_UNOBSERVABLE**: Observability matrix rank deficient; obstruction includes rank analysis.
3. **FILTER_DIVERGED**: Estimate drifted from truth; obstruction includes divergence bound.
4. **COVARIANCE_SINGULAR**: Covariance matrix degenerate.

8.2 Kalman Recompute Hook

The Kalman recompute hook verifies:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - H\hat{x}_{k|k-1})$$

using exact rational arithmetic.

9 RL Certificates (Chapter 12)

Definition 10 (RLCertificate)**.** *An RL certificate attests to reinforcement learning:*

$$\texttt{RLCert} = (A, R, Q_w, P)$$

where A is the algorithm, R is the reward specification, Q_w is the Q-value witness, and P is the method proof.

9.1 Distance Delta Reward (QA-Native)

The QA-native reward specification is *distance delta*:

$$r(s, a, s') = d_{\text{BFS}}(s, \text{goal}) - d_{\text{BFS}}(s', \text{goal})$$

Positive reward for moving closer; negative for moving away.

9.2 Q-Learning Recompute Hook

```
RecomputeQ(cert, transitions):
    for (s, a, r, s', Q_before, Q_after) in transitions:
        Q' = Q_before + alpha * (r + gamma * max_Q(s') - Q_before)
        if Q' != Q_after:
            return "mismatch"
    return "verified"
```

10 Imitation Certificates (Chapter 13)

Definition 11 (ImitationCertificate)**.** *An imitation certificate attests to learning from demonstrations:*

$$\texttt{ImitCert} = (M, D_w, \texttt{IRL}_w, P)$$

where M is the imitation method, D_w is the demonstration witness, $\texttt{IRL}_w$ is the inverse RL witness, and P is the method proof.

10.1 IRL as Target-Class Inference (QA-Native)

Definition 12 (Inverse RL Witness)**.**

$$\texttt{IRLWitness} = (T_{inferred}, c, \texttt{id}, \{T_i\}_{alt})$$

where $T_{inferred}$ is the inferred target class, c is confidence, $\texttt{id}$ is identifiability flag, and $\{T_i\}_{alt}$ are alternative targets (if non-identifiable).

QA-Native Insight Inverse RL reduces to *target-class inference*: given demonstrated trajectories, infer which target class the expert was pursuing. This reuses the identifiability machinery from observer upgrade theorems.

11 Cross-Certificate Coherence

Certificates do not exist in isolation. A `CertificateBundle` groups related certificates and validates coherence.

11.1 Coherence Rules

1. **RL $\leftrightarrow$ Policy**: Generator sets must be compatible; RL average steps $\geq$ policy optimal distance.
2. **Imitation $\leftrightarrow$ Exploration**: Demo coverage should align with exploration coverage.
3. **Filter $\leftrightarrow$ Inference**: State observability relates to inference identifiability.
4. **MCTS $\leftrightarrow$ Exploration**: Exploration methods and UCB constants should be consistent.

11.2 Bundle Manifest

Each bundle produces a cryptographic manifest:

$$\texttt{Manifest} = (\texttt{id}, \texttt{SHA256}(B), n, \{c_i\})$$

enabling tamper-evident certificate storage.

12 Implementation and Evaluation

12.1 Test Suite

The implementation includes 295 test cases covering:

- All seven certificate types
- All failure modes with obstruction generation
- Recompute hook verification
- Cross-certificate coherence validation

12.2 End-to-End Demo

The spine demo runs a 5×5 gridworld through all seven layers:

```
Planning (BFS optimal, 8 steps)
    |
MCTS (UCB1, 60% pruning via SCC)
    |
Exploration (UCB1, regret=50)
    |
Inference (VE, P(Goal|position)=1)
    |
Filtering (Kalman, x~4, y~4)
    |
RL (Q-learning, distance_delta, 95%)
    |
Imitation (IRL, 98% confidence)
```

All certificates pass validation; bundle coherence confirmed.

13 Discussion

13.1 The Failure-Completeness Theorem

We can now state the central theoretical contribution:

Theorem 13 (Failure-Completeness)**.** *For every decision process $\mathcal{D}$ admitted by the certificate spine, and every task specification T, exactly one of the following holds:*

1. *A **success certificate** exists with a verifiable witness proving task completion, or*

2. *A **failure certificate** exists with constructive obstruction evidence proving task impossibility.*

Proof sketch. The proof proceeds by structural induction over the seven certificate types. Each certificate validator is a total function that either: (a) accepts with witness, or (b) rejects with obstruction. The validators are implemented as terminating decision procedures with exact arithmetic, ensuring decidability. The obstruction types form a finite, exhaustive enumeration of failure modes for each layer. □

This theorem distinguishes the QA approach from standard benchmarks: failures are not error codes or missing data—they are *constructive proofs of impossibility*.

13.2 Failure as First-Class Object

The central insight is that failures should have the same mathematical status as successes. An obstruction is not an error code—it is a constructive proof that the task was impossible.

This has practical implications:

- **Debugging**: Obstruction evidence explains *why* a system failed, not just *that* it failed.
- **Safety**: Systems can prove they cannot reach dangerous states (obstruction certificate for bad outcomes).
- **Auditability**: Every decision has a machine-checkable justification.

13.3 Exact Arithmetic

Using $\mathbb{Q}$ instead of floating-point eliminates a class of validation ambiguities. When a recompute hook reports "verified," there is no ϵ-tolerance involved.

13.4 Limitations

- Exact arithmetic scales poorly to large factor graphs (rational explosion).
- SCC pruning requires explicit state graph (not applicable to continuous/high-dimensional spaces).
- Current implementation is single-threaded.

14 Conclusion

The QA Decision Certificate Spine provides a unified, auditable framework for sequential decision making. By treating failure as a first-class mathematical object, we enable a new paradigm: systems that can explain not just what they did, but why alternatives were impossible.

The 295-test implementation demonstrates feasibility. The path forward includes formal verification bridges (Lean/TLA+), external benchmark integration, and temporal extensions.

References

[1] G. Katz et al., "Reluplex: An efficient SMT solver for verifying deep neural networks," CAV 2017.

[2] E. Altman, *Constrained Markov Decision Processes*, CRC Press, 1999.

[3] A. Strehl et al., "Reinforcement learning in finite MDPs: PAC analysis," JMLR, 2009.

[4] M. Kochenderfer et al., *Algorithms for Decision Making*, MIT Press, 2022.