Algebraic Foundations of Deterministic Safety Certification

From Mathematical Invariants to Production Infrastructure

Author: William Tennant, Founder — Northbeam Solutions LLC

Organization: Northbeam Solutions LLC (northbeam.solutions)

Product: QAE Safety Kernel (qaesubstrate.com)

Date: March 2026

Abstract

The QAE Safety Kernel certifies autonomous actions before execution — deterministically, reproducibly, and across any domain. Prior white papers in this series have demonstrated the kernel's architecture, its application to financial risk and agentic AI, and a practical guide for building custom domain adapters.

This paper addresses a deeper question: *why does this work?*

We present the algebraic foundations that make domain-agnostic safety certification mathematically possible. The key insight is that safety certification reduces to a small set of algebraic operations — state space projection, parallel constraint evaluation, and conservative aggregation — that are invariant to what the state dimensions represent. A constraint margin is a constraint margin whether it measures portfolio Value-at-Risk, satellite orbital decay, or an AI agent's budget utilization.

We trace the mathematical lineage from quotient space theory through constraint algebra to the certification pipeline, showing that domain-agnosticism is not a design aspiration but a structural property of the algebra. We then connect these foundations to the production platform — the end-to-end infrastructure that transforms mathematical certification into an operational safety layer for any autonomous system at any scale.

The result is a unified framework where a single mathematical proof of correctness covers every domain adapter, every constraint channel, and every certification decision — past, present, and future.

---

1. Why Foundations Matter

There is a difference between a system that *happens* to work across multiple domains and one that is *mathematically guaranteed* to work across any domain. The distinction matters for three audiences:

For regulators and auditors: If domain-agnosticism is an engineering coincidence, it must be re-validated for each new domain. If it is a mathematical property, a single proof covers all domains — including domains that don't exist yet. This is the difference between certifying a product and certifying a *framework*.

For engineers evaluating build-vs-buy: A system with deep mathematical foundations ages differently than one built on heuristics. Heuristic systems accumulate technical debt as edge cases multiply. Algebraic systems remain correct as domains are added because the algebra doesn't change.

For investors assessing defensibility: Mathematical foundations create structural moats. A competitor can replicate an API; they cannot replicate a mathematical framework without independently deriving the same algebra. The IP protection is intrinsic to the depth of the mathematical contribution.

This paper makes the foundations explicit.

---

2. The Algebraic Framework

2.1 Quotient Mapping: Collapsing Domain Complexity

Every autonomous system operates in a domain-specific reality. A financial portfolio has positions, Greeks, sector allocations, and counterparty exposures. A satellite constellation has orbital parameters, thermal profiles, spectrum assignments, and collision geometries. An AI agent has budget utilization, rate consumption, scope boundaries, and data classification levels.

These realities are wildly different. A naive approach would build a separate safety system for each domain, hardcoding domain knowledge into every layer. This is how the industry currently operates — and it is why safety systems are expensive, fragile, and impossible to validate across domains.

The QAE approach begins with a different question: *what is the minimum representation of a domain state that preserves all safety-relevant information?*

The answer is a **quotient mapping** — a mathematical projection from a high-dimensional domain-specific state space to a compact numerical vector where each dimension corresponds to one safety-relevant quantity.

Formally: let D be the full domain state (positions, parameters, configurations — potentially thousands of dimensions). The quotient mapping $\phi: D \to R^n$ projects D into an n-dimensional state vector where n is the number of independent safety-relevant dimensions. The projection must satisfy:

1. Safety-preserving: If two domain states d_1 and d_2 map to the same state vector ($\phi(d_1) = \phi(d_2)$), they must have identical safety profiles. No safety-relevant distinction is lost.

2. Compact: The dimension n is minimal — typically 5-15 dimensions per domain, regardless of the domain's intrinsic complexity.

3. Deterministic: The mapping is a pure function. Same domain state, same vector, every time.

The quotient mapping is where domain knowledge enters the system — and the *only* place it enters. Everything downstream operates on the numerical vector, with no knowledge of what the dimensions represent.

Domain	Full State Complexity	Quotient Dims	Compression
Financial Risk	~100+ positions, Greeks, exposures	10	>10x
Agentic AI	API calls, tool permissions, data access	7	Variable
Satellite Constellation	Orbital mechanics, thermal, spectrum	15	>100x
Healthcare (reference)	Patient history, labs, medications	8-12	>1000x

The mathematical guarantee: once a valid quotient mapping exists for a domain, the kernel's certification is guaranteed correct for that domain. The proof obligation falls entirely on the mapping — not the kernel.

2.2 Constraint Channels: Parallel Safety Evaluation

With the state vector in hand, the kernel evaluates safety through independent **constraint channels**. Each channel is a function $c: R^n \to [0, 1]$ that maps the state vector to a scalar margin between 0 and 1, where 1.0 = maximum headroom (well within safe limits) and 0.0 = at the boundary (limit fully consumed).

Each channel encapsulates one dimension of safety. The channels are independent and evaluated in parallel. The critical algebraic property: **every channel conforms to the same signature**. The kernel doesn't know — and doesn't need to know — whether a margin of 0.35 represents remaining liquidity buffer or remaining satellite propellant. A margin is a margin. This is not an abstraction convenience; it is the mathematical foundation that makes domain-agnosticism possible.

The uniform [0,1] output range normalizes fundamentally incomparable quantities (dollars, percentages, orbital distances, classification levels) to a common scale where aggregation is meaningful. The normalization happens inside each channel. Outside the channel, only the margin exists.

2.3 Conservative Aggregation: The Bottleneck Principle

Given margins from all constraint channels, the kernel must produce a single aggregate safety assessment. The aggregation function must satisfy three properties:

Conservatism: The aggregate margin must never exceed the minimum channel margin. A system that is dangerous on one dimension is dangerous overall, regardless of how safe it is on other dimensions. Mathematically: $\text{aggregate}(m_1, ..., m_k) \leq \min(m_1, ..., m_k)$.

Monotonicity: If any channel's margin decreases, the aggregate margin cannot increase. Safety cannot improve when a constraint becomes tighter.

Identification: The aggregation must identify which channel is the bottleneck — the binding constraint that determines the overall safety level.

The QAE kernel uses a smoothed variant of the minimum that preserves all three properties while adding a fourth: **differentiability**. The smooth approximation lies strictly below the true minimum, maintaining conservatism, while providing gradient information that reveals how close each channel is to becoming the new bottleneck. This gradient is the foundation for time-to-breach estimation and drift monitoring.

The mathematical guarantee: because the aggregation is conservative, the kernel never certifies an action as safe when any constraint channel says it is dangerous. This is a provable property, not a test result.

2.4 Zone Classification: From Continuous Margin to Discrete Decision

The aggregate margin maps to a discrete safety zone through fixed thresholds:

Aggregate Margin	Zone	Decision
> 0.6	Safe	Certified - proceed
(0.3, 0.6]	Caution	Certified with Warning - proceed with monitoring
(0.1, 0.3]	Danger	Escalate to Human - do not proceed without review
≤ 0.1	Critical	Blocked - do not proceed

One override exists: **regime change detection**. When the domain adapter detects a fundamental shift in operating conditions (market regime change, solar storm, model version change), the kernel escalates to human review regardless of current margins. Margin calculations are valid within a given operating regime. When the regime changes, margins need re-calibration before they can be trusted.

2.5 The Composition Theorem

The full certification pipeline composes these operations:

```
certify(action, domain) =
    zone(aggregate(c1(phi(action)), c2(phi(action)), ..., ck(phi(action))))
```

The domain-agnosticism theorem: The functions *aggregate* and *zone* contain no domain-specific parameters. They operate identically regardless of what ϕ and $c_1 ... c_k$ compute internally. Therefore, any proof about *aggregate* and *zone* (conservatism, monotonicity, determinism) applies to all domains simultaneously.

This is not an empirical observation. It is a structural property of function composition. The kernel is domain-agnostic for the same reason that addition is domain-agnostic: 3 + 4 = 7 whether the numbers represent apples, dollars, or orbital velocities.

What must be proven per domain: Only the quotient mapping ϕ and the constraint channels $c_1 ... c_k$ require domain-specific validation: (1) ϕ must be safety-preserving, (2) each c_i must return values in [0,1], (3) each c_i must be monotone, (4) the channels must collectively cover all safety-relevant dimensions. Once established, the kernel's correctness follows by composition.

---

3. Determinism as a Mathematical Property

3.1 Why Determinism Matters

A safety certification system that produces different results on different runs — even if all results are "reasonable" — is fundamentally unsuitable for regulatory compliance, forensic investigation, or insurance underwriting. If an adverse event occurs and the safety evaluation cannot be reproduced exactly, the certification is worthless as evidence.

The QAE kernel is deterministic: the same input always produces the same output, bit-for-bit, across machines, operating systems, and time. This is not merely a testing convenience — it is a mathematical property enforced at every layer.

3.2 Sources of Non-Determinism and Their Elimination

Hash map iteration order. Standard hash maps use randomized hashing, producing different iteration orders across runs. The kernel uses ordered maps exclusively (lexicographic ordering by key), ensuring that every traversal is identical.

Floating-point representation. The same floating-point value can be formatted as "0.35", "3.5e-1", or "0.350000000000000" depending on the formatting function. The kernel uses IEEE 754 16-digit scientific notation for all floating-point-to-string conversions, producing a single canonical representation for every value.

Random number generation. The kernel uses no random sampling in any computation. Constraint evaluation, aggregation, and zone classification are pure functions of their inputs.

System clock. Timestamps appear only as metadata (when was the certificate issued), never as inputs to computation. Two certificates generated at different times but with identical inputs will have identical margins, zones, and decisions.

Thread scheduling. Constraint channels are evaluated independently and their results aggregated deterministically. The order of evaluation doesn't affect the result because the aggregation function is commutative and the results are stored in ordered collections.

3.3 The Hash Chain: Tamper-Evident Certification

Every safety certificate includes a SHA-256 hash computed from the certificate's contents in canonical form. The hash input is constructed from: (1) all channel margins in lexicographic order by channel name, (2) the aggregate margin, (3) the safety zone, (4) the decision, (5) the domain name, and (6) any regime change indicator.

Because every component is deterministic, the hash is deterministic. This creates a **tamper-evident audit trail**. Certificates can be stored, transmitted, and verified independently. A regulator can recompute any certificate's hash from its fields and confirm it hasn't been modified. An insurer can verify that the safety evaluation that preceded an action matches the logged evaluation.

The mathematical guarantee: given the same inputs, the kernel will produce the same certificate with the same hash, today, next year, or a decade from now — on any machine running the same kernel version.

---

4. Cross-Domain Proof of Generality

4.1 Three Domains, One Algorithm

The strongest evidence for domain-agnosticism is demonstration across maximally different domains. The QAE Safety Kernel has been implemented and tested across three domains that share no subject-matter overlap:

Financial Risk Management. 10-dimensional state vector. 5 constraint channels spanning market risk (VaR, DV01, FX exposure, commodity beta), credit risk (CS01, expected loss), liquidity risk (position-to-volume ratios, illiquid concentration), concentration risk (issuer and sector HHI, single-name limits), and regulatory capital (Basel III risk-weighted assets vs. available capital). Regime detection: portfolio state changes exceeding a threshold norm.

Autonomous AI Agent Governance. 7-dimensional state vector. 5 constraint channels spanning budget utilization (API spend vs. limit), rate limiting (actions per time window), scope enforcement (authorized action boundaries), reversibility assessment (undo capability of proposed action), and data sensitivity classification (PII exposure level). Regime detection: budget exhaustion below 10% remaining or rate exceeding 90% of limit.

Satellite Constellation Operations. 15-dimensional state vector. 5 constraint channels spanning spectrum coordination (bandwidth allocation and interference margins), thermal budget (heat dissipation vs. capacity), coverage capacity (active satellites vs. minimum required), collision avoidance (miss distance vs. safety threshold), and orbit decay (propellant reserves and orbital mechanics). Regime detection: solar storm thermal shifts, conjunction clusters, coverage crises.

4.2 What Is Identical Across Domains

The certification pipeline is literally the same function call across all three domains. The kernel code path does not branch on domain type. There is no switch statement, no domain-specific conditional, no special case. What is identical:

- The aggregation function (conservative minimum with smooth approximation)
- The zone classification thresholds and decision logic
- The certificate structure (margins, zone, decision, hash)
- The hash computation (canonical form, SHA-256)
- The determinism guarantees (ordered collections, canonical floating-point, no randomness)
- The audit trail format (append-only, hash-chained)

4.3 What Varies Across Domains

Only the adapter varies: the quotient mapping (what dimensions matter), the constraint formulas (domain-specific safety logic), the regime detection criteria (what constitutes a fundamental operating shift), and the state vector dimensionality (7, 10, 15 — or any other number).

This separation is enforced at the type system level: the kernel crate has zero dependencies on any domain crate. It cannot import domain-specific types, cannot reference domain-specific constants, and cannot conditionally compile domain-specific code. The adapter trait is the only interface, and it traffics exclusively in domain-agnostic types (floating-point vectors, string maps, boolean flags).

4.4 The Fourth Domain Test

The ultimate test of domain-agnosticism is a domain the kernel was never designed for. In our technical guide "Build Your Own Domain Adapter," we walk through Healthcare Clinical Decision Support as a worked example — an entirely new domain with constraint channels for drug interaction severity, dosage safety margins, contraindication coverage, lab value compliance, and patient allergy risk.

The exercise requires zero modifications to the kernel. This is the mathematical claim made concrete: domain-agnosticism is not a feature we built. It is a property of the algebra.

---

5. From Theory to Production: The End-to-End Platform

Mathematical correctness is necessary but not sufficient for production safety certification. The QAE platform wraps the algebraic kernel in production infrastructure that addresses the operational requirements of real deployments.

5.1 The Certification API

The kernel is accessible through a REST API that accepts a proposed action and returns a safety certificate. The API layer adds: **request validation** (domain-specific schemas), **response normalization** (opaque field names revealing no implementation details), **rate limiting** (tiered access: free, professional, enterprise), and **circuit breaking** (automatic degradation under failure conditions).

5.2 Real-Time Streaming

For domains where the operating state changes continuously (financial markets, satellite telemetry, agent activity), the platform provides WebSocket streaming with continuous state ingestion, automatic re-certification when state changes exceed drift thresholds, zone transition alerts, and deterministic cross-validation.

5.3 The Audit Trail

Every certificate is appended to a hash-chained audit log providing append-only integrity, independent verification, search by time/zone/decision/constraint/domain, and pluggable sinks (local storage, cloud object stores, compliance-grade archival).

5.4 The Developer Portal

JWT-based authentication with API key management, rotation, and grace periods. Real-time usage tracking. Metered billing with tiered pricing. Interactive API reference with request/response examples for each supported domain.

5.5 Domain Adapter SDK

For organizations building custom domains: **Rust SDK** (published crate with compile-time guarantees), **Python bindings** (PyO3-based, rapid prototyping), **example adapters** (all three production domains

plus a custom domain template), and **testing harness** (deterministic fixtures verifying margin bounds, monotonicity, and determinism).

5.6 The Operational Value Chain

```
Domain Expert -> Quotient Mapping + Constraints -> Adapter SDK
     |
Adapter -> QAE Safety Kernel -> Certification Pipeline
     |
Certificate -> Audit Trail -> Compliance Archive
     |
API -> Rate Limiting -> Usage Tracking -> Billing
     |
Dashboard -> Real-Time Monitoring -> Zone Alerts -> Human Escalation
```

Each layer adds operational value without modifying the mathematical core. The kernel remains a pure function: same inputs, same outputs, same hash.

---

6. The Containment Connection

Recent research has established a surprising connection between hard structural boundaries in computational systems and the density of safety-relevant information those systems can maintain.

The published finding (Tennant, 2026): imposing topological boundaries on a neural network's representational space — hard constraints that the optimization process cannot violate — increases Fisher information density by up to 104% without degrading task performance.

The connection to safety certification: the QAE kernel's constraint channels impose exactly this kind of hard structural boundary on autonomous system behavior. The containment research suggests that this kind of hard boundary doesn't just prevent bad outcomes — it may actively improve the quality of the decisions that remain within bounds.

The principle: **external-substrate constraint** — safety boundaries imposed from outside a computational process can increase the information-theoretic efficiency of the process they constrain. The safety system isn't just a cost center. It is a structural enhancement.

This finding is published and available for independent verification (DOI: 10.6084/m9.figshare.31742857).

---

7. Mathematical Properties Summary

The following properties are provable from the algebraic framework and hold across all domains:

Property	Statement	Proof Basis
Conservatism	Kernel never certifies safe when any channel reports danger	Aggregate <= min(margins), zone is monotone

Property	Statement	Proof Basis
Determinism	Same inputs produce identical certificates on any machine	Ordered collections, canonical representations, no randomness
Tamper evidence	Any modification to a certificate is detectable	SHA-256 hash of canonical form
Domain agnosticism	Kernel proofs apply to all domains simultaneously	Zero domain dependencies; operates on numerical vectors only
Composability	Adding a new domain requires no kernel modification	Adapter trait is only interface; no domain conditionals
Monotonicity	Worsening any margin cannot improve decision	Aggregate and zone classification are both monotone
Completeness	Every proposed action receives a definite decision	Pipeline is total: produces one of four decisions
Regime safety	Regime changes trigger mandatory escalation	Regime override precedes zone classification

These properties are enforced at the type system level (compile-time) and verified by 144+ tests across all domain adapters.

---

8. Implications for Autonomous System Governance

8.1 Regulatory Frameworks

EU AI Act (2024). High-risk AI systems must demonstrate conformity assessment. A domain-agnostic certification kernel with provable mathematical properties provides a natural foundation: certify once (the kernel's algebra), validate per domain (the adapter's quotient mapping and constraints).

NIST AI Risk Management Framework. The NIST AI RMF 1.0 calls for "measurable" and "regularly monitored" AI risk. The kernel's continuous [0,1] margins, real-time monitoring, and deterministic audit trail satisfy these requirements by construction.

SEC Staff Statements on AI in Securities Markets (2024). Broker-dealers and investment advisers using AI must demonstrate risk management. Deterministic safety certification with tamper-evident audit trails provides the evidence base that compliance officers need.

ISO/IEC 42001 (AI Management System). The standard requires organizations to "manage risks related to AI systems." A certified-before-execution model with per-action audit trails maps directly to ISO 42001's control objectives.

The mathematical framework provides a path to **framework-level certification**: instead of certifying each domain deployment independently, regulators can validate the algebraic framework once and then validate only the domain-specific adapter for each new deployment. This dramatically reduces the compliance cost of deploying autonomous safety across new domains.

8.2 Insurance and Liability

Insurance underwriters need quantifiable risk metrics and reproducible evidence. **Pre-event:** continuous margin monitoring with configurable alerting thresholds. **Post-event:** deterministic certificates with tamper-evident hashes provide forensic evidence. **Actuarial modeling:** the [0,1]

margin scale enables actuarial models that relate margin levels to loss frequencies.

8.3 Cross-Domain Safety Standards

If the algebra is domain-agnostic, then safety zones (Safe, Caution, Danger, Critical) have consistent meaning across domains. A margin of 0.35 means the same thing — relative distance from constraint boundaries — whether the domain is finance, healthcare, or aerospace. This enables transferable safety auditing skills, unified safety dashboards, and industry benchmark development on a shared mathematical foundation.

---

9. Conclusion

The QAE Safety Kernel is not a collection of safety checks. It is a mathematical framework with provable properties that hold across any domain.

The algebraic foundations — quotient mapping, parallel constraint evaluation, conservative aggregation, monotone zone classification, and deterministic hash-chained certification — are not implementation details. They are the invariants that guarantee the system's correctness. When a new domain adapter is connected, the kernel's existing proofs apply immediately. No re-validation of the kernel is required.

For organizations deploying autonomous systems, the implication is clear: safety certification can be infrastructure. Not a per-domain cost center. Not a set of ad-hoc checks. Infrastructure — with the mathematical guarantee that it works today, will work tomorrow, and will work for domains that haven't been imagined yet.

The algebra doesn't care what the dimensions represent. That is the point.

Appendix A: Trait Signatures (Rust)

The kernel's domain-agnostic interface is defined by two traits:

```
/// Domain adapter: bridges domain-specific logic to the kernel
pub trait DomainAdapter: Send + Sync {
    fn domain_name(&self) -> &str;
    fn constraint_channels(&self) -> Vec<Box<dyn ConstraintChannel>>;
    fn map_action_to_state(
        &self, action: &dyn ProposedAction,
    ) -> KernelResult<Vec<f64>>;
    fn detect_regime_change(
        &self, current: &[f64], proposed: &[f64],
    ) -> bool;
    fn format_domain_payload(
        &self, margins: &BTreeMap<String, f64>,
    ) -> Option<serde_json::Value>;

    fn current_state(&self) -> KernelResult<Vec<f64>> {
        Err(KernelError::AdapterError(
            "current_state not implemented".into(),
        ))
    }
}

/// Constraint channel: one dimension of safety evaluation
pub trait ConstraintChannel: Send + Sync {
    fn name(&self) -> &str;
    fn evaluate(&self, state: &[f64]) -> KernelResult<f64>;
    fn dimension_names(&self) -> Vec<String>;
}
```

These two traits are the *complete* interface between the kernel and any domain. The kernel imports nothing else from domain code.

Appendix B: Certification Pipeline (Pseudocode)

```
function certify_action(adapter, action, config):
    // Step 1: Quotient mapping
    state = adapter.map_action_to_state(action)

    // Step 2: Parallel constraint evaluation
    margins = {}
    for channel in adapter.constraint_channels():
        margins[channel.name()] = channel.evaluate(state)

    // Step 3: Conservative aggregation
    binding_margin = min(margins.values())
    binding_channel = argmin(margins)

    // Step 4: Regime detection
    current = adapter.current_state()
    regime_change = adapter.detect_regime_change(current, state)

    // Step 5: Decision logic
    if regime_change:
        decision = EscalateToHuman
    else if binding_margin > 0.6:
        decision = Certified
    else if binding_margin > 0.3:
```

```
    decision = CertifiedWithWarning
else if binding_margin > 0.1:
    decision = EscalateToHuman
else:
    decision = Blocked

// Step 6: Certificate generation
return SafetyCertificate {
    margins, binding_channel, binding_margin,
    zone: classify_zone(binding_margin),
    decision,
    hash: sha256(canonical_form(...))
}
```

Appendix C: Cross-Domain Channel Analogies

Safety Function	Finance	Agentic AI	Constellation
Primary risk metric	Market risk (VaR, DV01)	Budget utilization	Spectrum coordination
Secondary risk metric	Credit risk (CS01)	Rate limiting	Thermal budget
Capacity/availability	Liquidity risk	Scope enforcement	Coverage capacity
Concentration/conflict	Concentration risk (HHI)	Reversibility	Collision avoidance
Regulatory/lifetime	Regulatory capital (RWA)	Data sensitivity	Orbit decay

The analogy is structural: each row represents a category of safety concern that appears in every domain. The specific formulas differ completely, but the [0,1] margin output is uniform.

Appendix D: Further Reading

Resource	Description
Deterministic Safety Certification for Autonomous Systems	Architecture and motivation (qaesubstrate.com/whitepapers)
Build Your Own Domain Adapter	Technical implementation guide with Healthcare worked example
Agentic AI Safety Certification	Deep dive into AI agent governance
Financial Risk Certification	Deep dive into portfolio risk management
Containment as Catalyst for Representational Quality	Foundational research (DOI: 10.6084/m9.figshare.31742857)
QAE Safety Kernel SDK	pip install qae-safety / cargo add qae-safety-kernel

---

Contact: bill@northbeam.solutions | qaesubstrate.com