RESEARCH PREVIEW

The Rule Problem

Key Hazards in AI Boundary Enforcement

DifferentialAILab

Differential AI Lab · 2026

ABSTRACT

As AI systems move from narrow tools to autonomous agents operating within organizations, boundary enforcement becomes a critical reliability and compliance surface. It is at these boundaries where organizations need to constrain what an AI system can do. Reliable controls require deterministic enforcement, moving the failure mode from model behavior to the rules that govern it: how those rules are generated, how they are translated from organizational intent, and whether they hold at runtime. This paper identifies four classes of hazards across that chain and maps them to complementary mitigation strategies grounded in neuro-symbolic constraint, cryptographic gating, and behavioral proof.

1. Introduction

The AI safety conversation largely focuses on model behavior: alignment, refusal, jailbreaking. In practice, most enterprises do not deploy foundation models directly, they deploy agents and pipelines governed by rules. The safety profile of those systems is a function of how well those rules are constructed, maintained, and enforced, not just how well-aligned the underlying model is.

Boundary enforcement is the operational problem of ensuring that an AI system stays within its intended operational scope, semantically, behaviorally, and over time. Getting it wrong produces systems that appear compliant during evaluation and fail in deployment.

Three hazard classes account for the majority of enforcement failures we observe. They occur at different points in the rule lifecycle and require different remediation approaches.

2. Three Hazard Classes

H1: Lossy Policy Translation

Organizational policies are written in natural language by humans who carry enormous amounts of implicit context. A policy like 'do not share confidential customer data with external parties' contains unstated assumptions about what counts as confidential, who constitutes an external party, and which data modalities are in scope. When that policy is translated into machine-executable rules, whether manually or via an LLM, that context is almost always lost.

The resulting rules are structurally well-formed but contextually incomplete. They pass review because they look correct on the surface. They fail in deployment because they do not cover the

cases the policy author assumed were obvious. This is not an edge case problem. It is the default state of policy translation at scale.

Key failure modes:

- Implicit context collapse, rules omit conditions that were never made explicit in the source policy
- Temporal scope blindness, rules reflect policy intent at a point in time but do not encode update triggers
- Principal hierarchy ambiguity, rules do not capture who can override what, under which conditions

H2: Rule Generation Pathologies

LLM-generated rules introduce a distinct failure class. When an LLM is used to generate, expand, or audit rules, a technique used in advanced compliance workflows, it brings both capability and systematic bias. The output can be fluent, comprehensive-looking, and wrong in ways that are difficult to detect without formal verification.

The core problem is that LLMs optimize for surface coherence, not exhaustive correctness. A generated rule set may cover the obvious cases well and fail precisely on the boundary conditions that matter most, where a malicious or errant agent is most likely to operate.

Key failure modes:

- Boundary softening, generated rules use probabilistic language ('should', 'generally') that symbolic enforcement engines cannot evaluate deterministically
- Conflict blindness, rule sets generated incrementally accumulate contradictions that are not surfaced until runtime
- Capability horizon mismatch, rules are written against the rule author's model of what the AI can do, which may not reflect actual capability at time of execution
- Overfitting to example distribution, generated rules handle the cases the system was trained on well and generalize poorly to novel configurations

H3: Runtime Enforcement Gaps

Even well-constructed rules fail at runtime when the enforcement architecture cannot keep pace with the system it governs. Agentic systems are inherently stateful and concurrent. Rules evaluated at one point in time may be stale by the time the action they govern executes.

This is particularly acute in multi-agent architectures where multiple agents share state, compose outputs, or operate on overlapping resource scopes. Individually compliant agents can produce collectively non-compliant behavior, a composability failure that point-in-time rule evaluation does not detect.

Key failure modes:

- Evaluation-execution lag, rules are assessed at decision time; actions complete after state has changed
- Context window truncation, long-horizon agents lose early rule context; in-model enforcement decisions are made without full constraint set
- Composability violations, each agent's individual outputs are rule-compliant; their composition is not
- Adversarial state injection, inputs are crafted to shift system state into regions where enforced rules do not apply

H4: Complexity-Driven Deadlock and Evaluation Loops

Rule sets grow. Organizational policies expand, edge cases accumulate, new capabilities require new constraints. What begins as a tractable set of discrete rules becomes an interdependent graph, and graphs have cycles.

Rule deadlock occurs when two or more rules interact such that no compliant action exists. Each rule is individually valid; their intersection produces an empty solution space. The system cannot proceed without violating at least one constraint. In agentic systems, deadlock typically manifests as task abandonment or silent failure rather than an explicit error, the agent stops making progress without surfacing why.

Rule evaluation loops occur when the enforcement engine must evaluate Rule A to determine whether Rule B applies, and Rule B to determine whether Rule A applies. In static rule systems this is a design error caught at compile time. In dynamic systems, where rules are generated at runtime, composed across agents, or conditioned on live state, cycles emerge that no static analysis anticipated.

Complexity is the forcing function for both. A ten-rule system has 45 possible pairwise interactions. A hundred-rule system has 4,950, and a still small thousand rule system nearly half a million. The probability of conflict or cycle grows much faster than the rule set. This is not a failure mode that careful rule writing prevents at scale; it is a structural property of complex rule graphs.

Key failure modes:

- Mutual exclusion deadlock, two rules that are individually satisfiable but jointly unsatisfiable block task completion
- Priority inversion, a high-priority rule is blocked by a lower-priority rule it cannot override due to scope overlap
- Evaluation cycle, rules conditioned on system state that they themselves affect create oscillating enforcement decisions
- Cascade invalidation, satisfying one rule changes system state such that a previously satisfied rule is violated, triggering re-evaluation indefinitely
- Complexity cliff, rule set behavior is tractable below a threshold and becomes computationally intractable above it, with no visible transition boundary

3. Mitigation Architecture

No single technique addresses all four hazard classes. Robust enforcement requires a layered architecture where each layer targets specific failure modes.

Neuro-symbolic constraint engines

Neural flexibility is necessary for semantic understanding, parsing natural language policies, handling ambiguous inputs, operating in high-dimensional state spaces. Symbolic precision is necessary for deterministic enforcement, rules that produce the same output for the same input without exception. Neuro-symbolic engines combine both: neural components handle interpretation and classification; symbolic components enforce the invariants that must hold regardless of input.

This directly addresses H1 and H2. The symbolic layer cannot be softened by probabilistic language. Contradictions in the rule set are structurally detectable. Boundary conditions receive the same treatment as central cases.

Cryptographic gating mechanisms

Access controls that are mathematically enforced rather than policy-enforced cannot be circumvented by instruction manipulation, prompt injection, or adversarial state injection. Cryptographic gating binds specific capabilities to verifiable conditions, the capability is unavailable, not merely discouraged, when those conditions are not met.

This directly addresses H3. Runtime enforcement gaps exist when enforcement is advisory rather than structural. Cryptographic gating makes the gap structural: the action cannot execute without satisfying the gate condition, independent of the agent's internal state.

Behavioral proof techniques

Formal verification of AI system behavior, establishing mathematical guarantees about what a system can and cannot do across its full input space, addresses the coverage problem that testing cannot solve. Testing demonstrates that rules hold for tested inputs. Behavioral proofs demonstrate that rules hold for all inputs within a defined scope.

This addresses H2 and H4. For H2, formal methods establish the actual reachable behavior space and verify that the rule set fully constrains it. For H4, the same machinery can statically detect deadlock conditions and evaluation cycles in the rule graph before deployment, transforming a runtime failure into a design-time artifact. Satisfiability solvers and model checkers applied to the rule set can enumerate all pairwise and higher-order conflicts, identify cycles, and establish complexity bounds on rule evaluation. This maps and identifies safe and unsafe areas of the complexity landscape.

4. Open Problems

Several challenges remain open at the intersection of these approaches:

- Rule lifecycle management, the mechanisms for updating rules in response to policy changes, capability updates, and observed failure modes are underspecified in current practice.
- Composability proofs at scale, behavioral proofs are well-established for individual systems; proving composability properties for multi-agent systems at realistic scale remains an active research area.
- Human-in-the-loop enforcement, the appropriate boundary between automated enforcement and human review varies by risk tier and solution maturity. Evolving and maintaining sufficient oversight is a well known system control problem that historically has proven to require continuous organizational effort.
- Cross-organizational rule compatibility, as AI agents increasingly operate across organizational boundaries, the problem of rule interoperability becomes significant.
- Rule complexity governance, the thresholds at which rule sets transition from tractable to intractable are poorly understood; tooling to continuously monitor and surface this boundary in production is nascent.

5. Conclusion

Boundary enforcement failures are not primarily model problems. They are rule problems, failures in how behavioral constraints are constructed, translated, and maintained. The four hazard classes described here, lossy policy translation, rule generation pathologies, runtime enforcement gaps, and complexity-driven deadlock, account for the majority of enforcement failures observed in deployed agentic systems.

Addressing them requires purpose-built architecture: **neuro-symbolic engines** to maintain semantic precision, **cryptographic gating** to enforce structural constraints that cannot be manipulated away, and **behavioral proof techniques** to provide formal guarantees about system behavior at coverage levels testing cannot reach.

As organizations deploy AI agents with increasing autonomy and resource access, the quality of boundary enforcement becomes proportional to the risk exposure of the system. The rule problem is not a corner case, it is the central engineering challenge in safe agentic AI deployment.

About Differential AI Lab

 Differential AI Lab is an Applied AI Lab specializing in Organizational AI Transformation, AI Native Software Development, AI Compliance, and Agentic Boundary Enforcement. Our boundary research combines neuro-symbolic constraint engines, cryptographic access controls, and behavioral proof techniques to address the operational challenges of deploying AI systems at enterprise scale.
Find out more at https://difflab.ai