

Liability as Authority Flow:

A Structural Model of Responsibility in AI and Complex Systems

Rob Caswell
Cyber Terrain

Abstract

Liability is conventionally analyzed as a retrospective judgment applied after harm occurs. This framing obscures a more fundamental question: why was the harm structurally possible in the first place? In complex and AI-driven systems, where execution is distributed, automated, and rapid, this question becomes central.

This paper introduces a structural model of liability grounded in system architecture rather than doctrinal categories. It defines a minimal set of liability primitives—duty, breach, causation, harm, and attribution—and decomposes each into operational components. The paper then demonstrates that these components correspond directly to features of system design, including constraint boundaries, reachable execution paths, and consequence-bearing primitives.

The central contribution is a formal mapping between liability and what is termed “authority architecture”: the topology of who or what can effect consequential state transitions within a system. Under this model, breach corresponds to constraint failure within the authority surface, causation corresponds to realized authority flow, and harm corresponds to activation of consequence primitives. Attribution distributes responsibility based on structural alignment with these layers.

This reframing allows liability to be analyzed *ex ante* as a property of system structure, rather than solely *ex post* as a function of adjudication. The model is particularly relevant to AI systems, where agentic execution and tool integration expand authority surfaces in ways not captured by traditional governance approaches.

Keywords

Liability, Authority Architecture, AI Governance, Causation, Breach, Systems Theory, Attribution, Cybersecurity, Autonomous Systems

I. Introduction

Liability is typically treated as a retrospective judgment. Harm occurs, facts are assembled, and responsibility is assigned through doctrine. This sequence is well established in tort, contract, and regulatory law, but it obscures a more fundamental question: why was the harm structurally possible in the first place?

In modern systems, that question is no longer theoretical. Distributed infrastructure, automated execution, and AI-driven decision-making have transformed how consequences arise. Actions are no longer confined to discrete actors or bounded environments. Instead, they propagate across interconnected systems, often through paths that are neither linear nor immediately visible.

Publication Record

Liability as Authority Flow: A Structural Model of Responsibility in AI and Complex Systems

Author: Rob Caswell

Publisher: Cyber Terrain LLC

Published: 2026-01-01

Zenodo version: 1.0

DOI: <https://doi.org/10.5281/zenodo.21742472>

Website: <https://cyberterrain.ai/papers/liability-as-authority-flow.pdf>

Corpus Context

Corpus: Authority Engineering

Document ID: AE-RES-0004

Object family: AE-RES-0004

Publisher-imprint edition prepared for versioned republication.

The prior Zenodo record remains the citable record until its new version is published.

Traditional liability frameworks remain necessary, but they are not designed to describe these underlying structures. They explain how responsibility is assigned after harm occurs, not how system architecture makes that harm possible.

This paper advances a different perspective. It treats liability as the observable outcome of authority flow through insufficiently constrained systems. When authority—defined as the ability to effect consequential state change—is not properly bounded, it can propagate through reachable paths and produce harm. Liability, in this view, is the legal manifestation of that flow.

The central claim is straightforward:

Liability can be modeled as the result of authority flow through insufficiently constrained systems, where duty defines expected constraint boundaries, breach represents constraint failure, causation represents realized flow, harm represents consequence, and attribution distributes responsibility across actors aligned with those structures.

This approach does not replace doctrine. It provides a structural layer beneath it.

Transition: The next section examines the limits of existing liability frameworks in describing these structural conditions.

II. Limits of Existing Liability Frameworks

Existing liability analysis is doctrinally robust but structurally thin.

In negligence, liability turns on duty, breach, causation, and damages. In products liability, the focus shifts to defect and foreseeability. In regulatory regimes, liability may attach through statutory violation regardless of traditional fault structures. Across these frameworks, the analytical process remains largely ex post and fact-specific.

This orientation produces three limitations in modern systems:

Compression of System Structure

Complex system behavior is reduced to discrete doctrinal elements. Distributed control, multi-step execution, and cross-boundary propagation are often collapsed into simplified causal narratives.

Limited Visibility into Path Formation

Causation is typically treated as linkage rather than topology. Courts evaluate whether harm is sufficiently connected to conduct, but rarely analyze the structure of the path through which harm propagated.

Weak Ex Ante Predictive Power

Because liability is assessed after harm occurs, existing frameworks offer limited ability to identify latent liability conditions embedded in system design.

These limitations become more pronounced in AI systems, where authority is distributed across models, tools, platforms, and operators.

These limitations motivate the need for a structural model of liability grounded in system behavior rather than doctrinal abstraction.

III. A Minimal Structural Model of Liability

Liability can be expressed through a minimal set of primitives:

- Actor
- Duty
- Breach
- Causation
- Harm
- Attribution

These primitives are necessary and jointly sufficient for liability formation across domains.

A. Duty

Duty defines the expected constraint boundary. It operates as both a recognition gate (whether an obligation exists) and a calibration function (how demanding that obligation is).

B. Breach

Breach is not merely deviation from a rule. It is the failure of effective constraint relative to the duty.

Breach decomposes into five structural modes:

- absence of boundary
- weakness of boundary
- bypassability
- delay in enforcement
- misalignment between boundary and consequence space

These modes correspond to failures in system constraint architecture rather than isolated acts.

C. Causation

Causation describes the realized path from breach to harm. It is not simply linkage, but propagation through a system.

It can be characterized by:

- continuity
- directness
- dependency
- resistance to intervention
- temporal compression

Foreseeability operates as a limiting condition on causal extension.

D. Harm

Harm is a structured consequence object, not a scalar loss. Harm may also arise through representation-mediated effects, where systems induce reliance by presenting outputs as authoritative, even when no direct physical or financial execution occurs.

It includes:

- consequence type (financial, physical, identity, etc.)
- magnitude
- irreversibility
- breadth of propagation
- persistence over time
- protected-interest salience

E. Attribution

Attribution distributes liability across actors based on structural alignment with:

- duty ownership
- constraint control
- path governance
- consequence proximity

Representational ownership operates as a conditional amplifier in reliance-facing systems.

Transition: The next section formalizes how these primitives interact to produce liability.

IV. Liability Formation and Structure

Liability formation can be expressed structurally:

$$L_f = D_0 \cdot B \cdot C' \cdot H_0$$

Interpretation of Terms

Each variable represents a bounded scalar capturing the presence or magnitude of the corresponding structural condition within a system. The model admits both binary and continuous interpretations depending on context.

D_0 represents duty existence as a recognition condition (0 if no duty is present; 1 if duty exists).

B represents breach as the degree of constraint failure, reflecting absence, weakness, bypassability, delay, or misalignment of constraint boundaries.

C' represents causation as realized authority flow, capturing the continuity, directness, dependency, and resistance to intervention of the execution path.

H_0 represents the presence of harm as a consequence condition (0 if no harm occurs; 1 if harm is realized).

H_s represents harm severity, incorporating magnitude, irreversibility, breadth, persistence, and protected-interest salience.

$\alpha(a_i)$ represents the attribution function over actor a_i , distributing liability proportionally based on structural alignment with duty ownership, constraint control, path governance, and consequence proximity.

Where:

- duty exists
- constraint fails
- failure propagates
- harm occurs

Severity is then applied:

$$L^* = L_f \cdot H_s$$

And distributed across actors:

$$L_i = \alpha(a_i)L^*$$

Liability formation is conjunctive, consequence is scaled, and allocation is distributive across actors.

V. Authority Architecture

The structure described above corresponds directly to system design.

Complex systems can be modeled in terms of:

- what actions are possible
- who or what can execute them
- what constraints exist
- how consequences propagate

This can be formalized as authority architecture.

The authority surface of a system is the set of consequential state transitions that actors are structurally permitted to execute.

Within this framework:

- duty corresponds to declared constraint architecture
- breach corresponds to constraint failure
- causation corresponds to authority flow

- harm corresponds to consequence primitive activation
- attribution corresponds to authority alignment across system layers

The authority surface described here can be represented as a weighted directed graph in which actors form nodes and authority transitions form edges. Constraint primitives operate as conditions on edge activation or transformation. Under this representation, authority flow corresponds to path traversal across the graph, and liability formation corresponds to the existence of reachable paths from constraint failure to consequence primitives. This structure admits formalization using standard graph representations, including adjacency, incidence, degree, and Laplacian matrices, and is a candidate for formal verification in higher-order logic frameworks.

VI. Liability as Authority Flow

Liability can be understood as the scalar projection of authority flow through insufficiently constrained authority surfaces.

More concretely:

- breach identifies where constraints failed
- causation describes how authority moved through the system
- harm identifies the consequence primitive activated
- attribution distributes responsibility across actors aligned with those layers

This reframes liability from a purely normative assessment to a structural outcome.

Recent legal scholarship has identified a parallel category of structural risk arising from defects in model formation. Kahana describes this as “representational risk,” referring to errors embedded in the internal geometry of self-supervised models. While representational risk concerns what a system believes is possible, authority architecture concerns what a system is permitted to execute. A representational defect may generate unsafe candidate actions, but harm arises only when those actions are allowed to propagate through an unconstrained authority surface. Liability in AI systems therefore emerges from the interaction between representational and authority layers rather than from either in isolation.²¹

VI.A. Representational Risk and Authority Architecture

Recent legal scholarship has begun to identify a distinct category of structural liability arising from defects in model formation rather than system execution. Eran Kahana describes this as “representational risk,” referring to errors embedded in the representational geometry of self-supervised models during training.¹

Kahana’s analysis identifies a gap in traditional liability frameworks. Liability is typically assigned at the level of inputs (e.g., training data provenance) or outputs (e.g., harmful or inaccurate responses). The intermediate phase—self-supervised learning—remains largely unaddressed, despite being the stage at which structural defects may be encoded into the model’s internal representation of the world.²

This perspective complements, rather than replaces, the model presented in this paper. Where representational risk concerns the internal geometry of the model—its “map” of the world—

authority architecture concerns the external topology of the system—its capacity to act within that world.

The relationship can be understood as follows: representational geometry constrains what the system believes is possible, while authority architecture determines which of those possibilities can be executed as consequential state transitions. A defect in representation may generate erroneous or unsafe candidate actions, but harm arises only when those actions are permitted to propagate through an unconstrained authority surface.

Accordingly, representational risk and authority risk are distinct but interacting layers. The former is rooted in model formation; the latter in system design. Liability analysis that considers only one layer risks misidentifying both the source of failure and the appropriate point of intervention.

This distinction has direct implications for attribution. Representational defects may support claims grounded in negligent model development or training processes, while authority failures support claims grounded in system architecture, deployment design, and execution control. In AI systems that combine both defects, liability may arise from their interaction rather than from either layer in isolation.

VII. Case Applications: Liability as Authority Flow in Practice

The following cases demonstrate that liability outcomes align with patterns of constraint failure, authority flow, and consequence realization.

A. Target Data Breach (2013)

Facts (compressed)

Attackers accessed Target's network through a third-party HVAC vendor, moved laterally across internal systems, and exfiltrated payment card data affecting over 40 million customers.

Structural Analysis

Duty

Target owed a duty to protect customer payment data.

$$D_0 = 1$$

Breach (Constraint Failure)

The breach reflects multiple simultaneous failures in constraint architecture:

- Bypassability: vendor credentials provided an alternate access path
- Scope mismatch: perimeter controls did not govern internal lateral movement
- Weakness: insufficient segmentation and detection capability
- Delay: alerts were generated but not acted upon

$$B \rightarrow \text{high}$$

Causation (Authority Flow)

- Continuity: uninterrupted lateral movement across systems
- Dependency: breach directly enabled data exfiltration
- Intervention resistance: alerts failed to interrupt the path
- Temporal compression: attack progressed before effective response

$C' \rightarrow \text{high}$

Harm

- Type: financial and identity/data
- Magnitude: high
- Breadth: large-scale customer impact
- Persistence: extended downstream effects

$H_0 = 1, H_s \rightarrow \text{high}$

Attribution

Target aligns across all attribution dimensions:

- duty ownership: high
- constraint control: high
- path governance: high
- consequence proximity: high

$\alpha(\text{Target}) \rightarrow \text{dominant}$

Interpretation

Liability attaches because authority flowed through a network where constraint architecture failed to align with reachable internal consequence paths. The breach did not merely occur at the perimeter; it manifested as a systemic failure to constrain authority movement within the network.

B. Knight Capital (2012)

Facts (compressed)

A faulty deployment triggered uncontrolled trading activity, generating approximately \$440 million in losses within minutes.

Structural Analysis

Duty

Knight owed a duty to maintain safe and controlled trading systems.

$$D_0 = 1$$

Breach

- Absence: no effective kill switch
- Delay: inability to halt execution in real time
- Scope mismatch: deployment controls did not govern live execution behavior

$$B \rightarrow \text{extreme}$$

Causation

- Continuity: uninterrupted execution loop
- Directness: immediate transition from failure to market action
- Temporal compression: near-instant propagation
- Intervention resistance: system could not be stopped

$$C' \rightarrow \text{maximal}$$

Harm

- Type: financial and operational
- Magnitude: extreme
- Irreversibility: high once trades executed

$$H_s \rightarrow \text{extreme}$$

Attribution

Knight aligns fully across all dimensions:

- constraint control: maximal
- path governance: maximal
- consequence proximity: maximal

$$\alpha(\text{Knight}) \approx 1$$

Interpretation

This case reflects complete failure of constraint architecture within a highly compressed authority path. Once the system entered execution, no mechanism existed to interrupt or reverse authority flow.

C. Equifax Data Breach (2017)

Facts (compressed)

Failure to patch a known vulnerability allowed attackers to access sensitive data of over 140 million individuals.

Structural Analysis

Duty

Equifax owed a duty to protect consumer data.

Breach

- Absence: patch not applied
- Weakness: insufficient internal controls
- Delay: extended attacker dwell time
- Scope mismatch: inadequate segmentation

$B \rightarrow \text{high}$

Causation

- Continuity: sustained access over time
- Dependency: direct exploitation path
- Intervention resistance: lack of detection and interruption

$C' \rightarrow \text{high}$

Harm

- Type: identity/data
- Breadth: extreme
- Irreversibility: high
- Persistence: long-term

Attribution

Equifax dominates:

- duty ownership: maximal
- constraint control: maximal
- consequence proximity: maximal

Interpretation

Liability reflects prolonged authority flow enabled by both absence and delay in constraint enforcement. The harm's persistence and irreversibility amplify the consequences of that failure.

D. Uber Autonomous Vehicle (2018)

Facts (compressed)

An autonomous test vehicle struck and killed a pedestrian. The safety driver failed to intervene.

Structural Analysis

Duty

Uber owed a duty to operate test systems safely.

Breach

- Weakness: inadequate detection and response logic
- Delay: insufficient intervention timing
- Scope mismatch: system autonomy exceeded constraint coverage

$B \rightarrow \text{high}$

Causation

- Dependency: system failure central to outcome
- Path governance: human intervention point existed but failed
- Temporal compression: insufficient reaction window

$C' \rightarrow \text{high}$

Harm

- Type: physical
- Irreversibility: maximal
- Salience: maximal

Attribution (Split)

Uber:

- duty ownership: high
- constraint control: high
- path governance: high

Driver:

- path governance: moderate

- consequence proximity: high
 $\alpha(\text{Uber}) > \alpha(\text{Driver})$

Interpretation

Liability distributes across actors aligned with different layers of authority: system design and real-time path governance.

E. Air Canada Chatbot (2024)

Facts (compressed)

A chatbot provided incorrect refund information. The court held Air Canada responsible despite the chatbot's autonomous behavior.

Structural Analysis

Duty

Air Canada owed a duty regarding customer-facing representations.

Breach

- Scope mismatch: chatbot output not governed by policy
- Weakness: insufficient validation of responses

$B \rightarrow \text{moderate}$

Causation

- Directness: user relied directly on output
- Dependency: harm depended on representation

$C' \rightarrow \text{high}$

Harm

- Type: financial/reliance
- Magnitude: low
- Salience: moderate-high

Attribution

Air Canada dominates:

- duty ownership: high
- consequence proximity: high
- representational ownership: maximal

$\alpha(\text{Air Canada}) \rightarrow \text{dominant}$

Interpretation

Liability attaches because the organization owns the authority surface presented to the user, regardless of internal system composition.

VIII. AI-Native Case: Agentic Tool Execution and Unauthorized Financial Transfer

The model's predictive value is most visible in AI-native systems, where authority structure precedes doctrinal clarity.

Scenario (Composite)

An enterprise deploys an AI agent integrated with email, CRM, financial APIs, and document systems. The agent processes a maliciously crafted input, retrieves internal context, and executes a financial transfer through an API.

Structural Analysis

Duty

The organization owes a duty to constrain system execution and prevent unauthorized transactions.

$$D_0 = 1$$

Breach

- Scope mismatch: model constraints do not extend to tool execution
- Bypassability: prompt injection creates alternate execution path
- Weakness: insufficient validation between interpretation and action
- Delay: no real-time intervention

$$B \rightarrow \text{high}$$

Causation

- Continuity: uninterrupted execution path
- Directness: minimal separation between decision and action
- Dependency: transaction depends on agent output
- Intervention resistance: no gating control
- Temporal compression: machine-speed execution

$$C' \rightarrow \text{high}$$

Harm

- Type: financial
- Irreversibility: high
- Saliency: high

$$H_0 = 1$$

Attribution

Deployer:

- duty ownership: high
- constraint control: high
- path governance: high
- consequence proximity: high

Model Provider:

- partial constraint control
- limited path governance

$\alpha(\text{Deployer}) \rightarrow \text{dominant}$

Interpretation

Liability arises because the system exposed a continuous authority path from untrusted input to financial execution without constraint at the tool boundary. This scenario illustrates the distinction between representational and authority risk: even if the model's internal representation were flawed, the resulting harm depends on whether the system permits execution of those representations through its authority surface.²¹

Synthesis

In AI systems, liability emerges when authority surfaces expand to include consequential primitives without corresponding constraint architecture, allowing compressed paths from input to execution.

Key Statement

The risk does not arise from what the model knows, but from what the system allows it to do.

IX. Implications

This model yields four immediate implications:

1. Liability becomes structurally analyzable ex ante
2. Governance must focus on constraint architecture, not only compliance
3. AI liability requires analysis of authority surfaces and flow

4. Attribution must reflect system topology, not isolated actions

X. Conclusion

Liability doctrine describes responsibility after harm occurs. Authority architecture explains why harm was structurally possible.

As systems become more autonomous and interconnected, liability analysis that does not account for authority structure will systematically misidentify both risk and responsibility.

References

Legal Cases and Proceedings

Palsgraf v. Long Island Railroad Co., 248 N.Y. 339 (1928).

In re Target Corporation Customer Data Security Breach Litigation, 66 F. Supp. 3d 1154 (D. Minn. 2014).

Moffatt v. Air Canada, 2024 B.C.C.R.T. 149 (British Columbia Civil Resolution Tribunal).

Federal Trade Commission v. Equifax Inc., No. 1:19-cv-03297 (N.D. Ga. July 22, 2019).

Regulatory and Government Reports

National Transportation Safety Board. (2019). *Collision Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian, Tempe, Arizona, March 18, 2018* (NTSB/HAR-19/03).

U.S. Securities and Exchange Commission. (2013). *In the Matter of Knight Capital Americas LLC*, Exchange Act Release No. 70694.

Legal Frameworks and Restatements

American Law Institute. (1965). *Restatement (Second) of Torts*.

American Law Institute. (1998). *Restatement (Third) of Torts: Products Liability*.

Technical and Academic Sources

Eran Kahana, *The Foundry Problem: World Models and the Missing Liability Framework for Self-Supervised Learning* (CodeX, Stanford Law Sch., Mar. 6, 2026).

Saltzer, J. H., & Schroeder, M. D. (1975). *The protection of information in computer systems*. Proceedings of the IEEE, 63(9), 1278–1308.

National Institute of Standards and Technology. (2016). *Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems* (NIST SP 800-160).

National Institute of Standards and Technology. (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*.

Standards and Policy Frameworks

Organisation for Economic Co-operation and Development. (2019). *OECD Principles on Artificial Intelligence*.

OWASP Foundation. (2024). *OWASP Top 10 for Large Language Model Applications*.