

MACHINE-SPEED AUTHORITY AND THE COMPRESSION OF GOVERNANCE LAYERS

Rob Caswell

Framework Paper

March 2026

Authority Architecture © Rob Caswell

All rights reserved.

Publication Record

Machine-Speed Authority and the Compression of Governance Layers

Author: Rob Caswell

Publisher: Cyber Terrain LLC

Published: 2026-03-01

Zenodo version: v1.0

DOI: <https://doi.org/10.5281/zenodo.21742265>

Website: <https://cyberterrain.ai/papers/authority-dynamics.pdf>

Corpus Context

Corpus: Authority Engineering

Document ID: AE-DYN-0001

Object family: AE-DYN-0001

Publisher-imprint edition prepared for versioned republication.

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

Table of Contents

- 1. Abstract**
- 2. Keywords**
- 3. Executive Summary**
- 4. Introduction: Machine-Speed Authority**
- 5. From Authority Architecture to Authority Dynamics**
- 6. Contributions**
- 7. Authority Design Layers**
- 8. Authority Execution Layers**
- 9. The Authority Stack (7×7 Model)**
- 10. AI Traversal of Governance Layers**
- 11. Authority Layer Compression**
- 12. Basal Primitive Density and Congestion**
- 13. Foundational Decomposition**
- 14. Authority Propagation and Elastic Redistribution**
- 15. Control Path Compression and Governance Response**
- 16. Implications for AI Governance and Automated Systems**
- 17. Conclusion**
- 18. References**

1. Abstract

Authority within complex systems is typically structured as a set of layered governance mechanisms that translate institutional intent into operational action. These layers—ranging from policy and strategic objectives to technical controls and monitoring—are commonly understood to operate sequentially through human decision processes. However, automated and AI-driven systems alter the behavior of this structure by executing decisions at computational speed. In such environments, governance constraints embedded across multiple layers may be encountered simultaneously during system execution rather than sequentially through institutional processes.

This paper introduces the concept of Authority Dynamics to describe how authority behaves under conditions of automation and machine-speed decision-making. Building on the Authority Architecture framework established in prior work, the paper proposes a two-dimensional model—the Authority Stack—that represents the interaction between authority design layers and authority execution layers. Within this model, automated decisions traverse multiple governance constraints through machine-speed execution pathways.

Several dynamic phenomena emerge from this traversal behavior. These include authority layer compression, in which governance layers designed for sequential interpretation are encountered simultaneously during execution; basal primitive density, which describes the clustering of governance primitives near the operational base of the architecture; and basal congestion, where dense primitive interactions reduce representational separability. The paper also introduces foundational decomposition as a representational technique for preserving analytical clarity under conditions of high interaction density. Additional dynamics such as authority propagation paths, elastic authority redistribution, and control path compression explain how authority effects move across automated systems and why governance mechanisms must increasingly operate within machine-speed execution environments.

Together, these dynamics reveal that automation does not eliminate authority architecture but instead exposes its operational behavior. Understanding these dynamics is necessary for analyzing modern automated systems and for designing governance mechanisms capable of maintaining effective authority under conditions of large-scale automation and AI-driven decision-making.

2. Keywords: Authority Architecture; Authority Dynamics; AI Governance; Automated Systems; Machine-Speed Authority; Governance Layer Compression; Control Path Compression; Authority Propagation

3. Executive Summary

Authority in complex systems is typically structured through layered governance frameworks that translate institutional intent into operational decisions. Policies define goals, compliance mechanisms enforce constraints, operational procedures guide implementation, and technical systems execute actions. These governance layers are commonly assumed to operate sequentially through human decision processes.

Automated and AI-driven systems alter this structure. In automated environments, decisions are often executed within milliseconds, and governance constraints embedded across multiple layers may be encountered simultaneously during execution. As a result, authority does not simply pass sequentially through governance layers but is traversed at machine speed across the architecture of the system.

This paper introduces the concept of Authority Dynamics to describe how authority behaves under these conditions. Building on the Authority Architecture framework, the paper proposes the Authority Stack, a two-dimensional model representing the interaction between governance design layers and operational execution layers. Within this structure, automated systems traverse authority pathways that produce dynamic behaviors including governance layer compression, primitive density clustering in basal layers, authority propagation across execution environments, and elastic redistribution of governance pressures.

These dynamics have important implications for governance. Traditional oversight models often operate outside the execution path of automated systems and therefore may fail to intervene before decisions propagate through operational layers. Effective governance in automated environments increasingly requires control mechanisms embedded within execution systems and capable of operating at machine speed.

Understanding Authority Dynamics is therefore essential for designing governance frameworks capable of regulating modern automated infrastructures.

4. Introduction: Machine-Speed Authority

Authority in complex systems is typically designed as a layered structure. Policy establishes intent, strategy translates priorities, compliance constrains permissible action, operations define implementation, and technical controls govern execution. In institutional design, these layers are usually treated as sequential: authority is expected to move from higher-order governance commitments downward into practical execution, while oversight moves upward through reporting, auditing, and review.

Automated and AI-driven systems change the behavior of this structure. In such systems, authority does not simply pass through governance layers one at a time. Instead, multiple layers are traversed within the same execution event. A model inference, automated deployment, trading action, recommendation, or control decision can simultaneously instantiate policy assumptions, compliance constraints, technical controls, operational objectives, and real-world consequences. The result is not the disappearance of authority architecture, but its compression into machine-speed execution paths.

This paper argues that authority systems exhibit dynamic behavior under automation. Governance layers that are designed to operate sequentially may be traversed simultaneously during execution, producing observable phenomena such as layer compression, primitive density clustering, basal congestion, elastic redistribution, and control path shortening. These phenomena do not invalidate authority architecture. Rather, they reveal its dynamic properties under conditions of automated traversal.

Paper 1 established the architecture of authority by identifying where authority exists and how it is structured across systems. Paper 2 extends that work by analyzing how authority behaves when automated systems and AI operate across that architecture at machine speed. The focus here is not only on authority placement, but on authority motion, compression, propagation, and control.

To formalize these dynamics, this paper introduces a two-dimensional model of authority composed of design layers and execution layers. This model makes it possible to analyze how authority is structured, where it is executed, how

automated decisions traverse governance constraints, and why lower-layer primitive interactions become increasingly dense under AI load. It also provides a basis for explaining why traditional oversight mechanisms often fail when they operate more slowly than the systems they are meant to govern.

The central claim of this paper is that automation transforms authority from a predominantly sequential governance structure into a machine-speed dynamic system. Understanding this transformation is necessary for explaining how modern authority actually behaves and for developing governance systems capable of operating within automated environments.

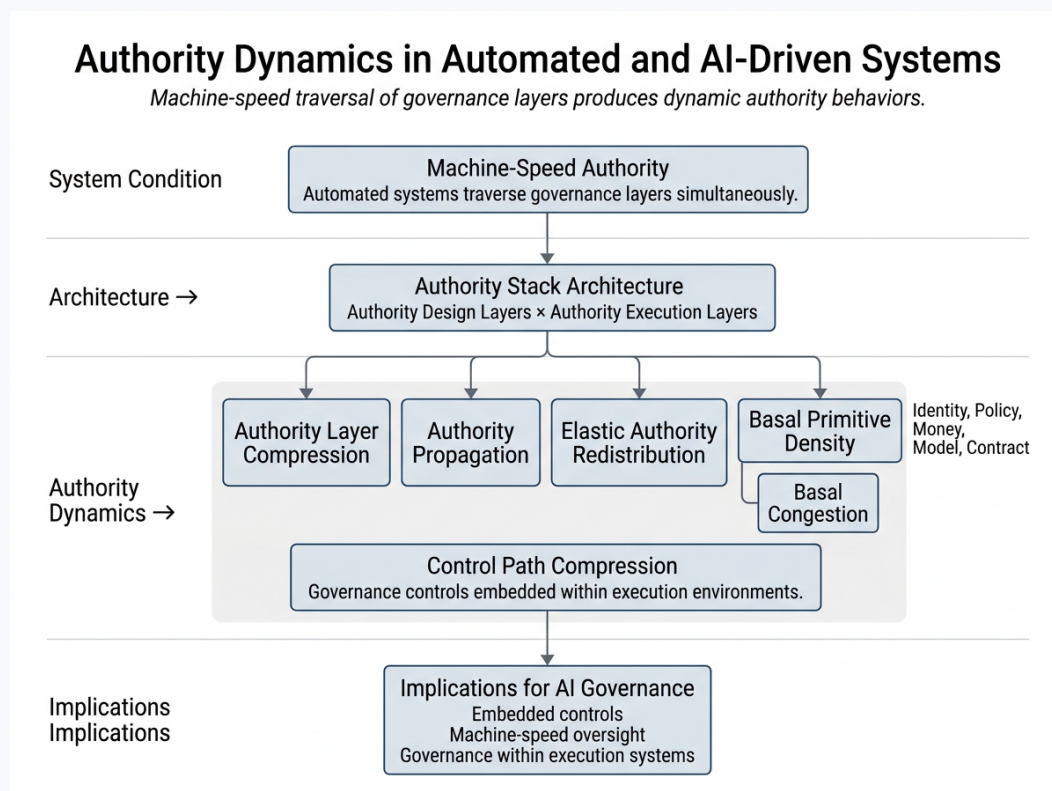


Figure 5. Authority Dynamics in Automated and AI-Driven Systems

5. From Authority Architecture to Authority Dynamics

Authority Architecture describes the structural placement of authority within a system. It explains where authority exists, how it is arranged, and how governance intent is distributed across components, actors, rules, and mechanisms. In that framework, authority is not reducible to formal hierarchy alone. It is embedded in policy, procedures, infrastructure, interfaces, constraints, incentives, and decision pathways.

That architectural view is necessary but incomplete for automated systems. A structural model can identify where authority is located, but it does not fully explain how authority behaves when system activity occurs at computational speed. In AI-enabled environments, authority is not merely stored in layered governance arrangements; it is activated, traversed, and propagated through them in real time.

Authority Dynamics is the study of this behavior. If Authority Architecture describes the static arrangement of authority, Authority Dynamics describes the temporal and operational properties of authority under execution. It examines how authority moves across layers, how governance constraints are compressed during automated action, how primitive interactions accumulate in lower layers, and how pressure redistributes when control mechanisms cannot keep pace with execution.

The relationship between the two papers is therefore direct. Paper 1 defines the architecture. Paper 2 explains the behavior of that architecture under load, automation, and AI traversal. The architecture remains intact, but its dynamic properties become visible only when systems execute fast enough to compress the governance layers that were designed to remain distinct.

This distinction matters because many governance failures in automated systems are misdiagnosed as failures of policy, ethics, or compliance alone. In many cases, the deeper problem is dynamic: governance layers exist, but the execution path crosses them too quickly, too densely, or too simultaneously for conventional controls to operate as intended. Authority Dynamics provides a framework for analyzing that condition.

6. Contributions

This paper makes four primary contributions to the study of authority and governance in automated systems.

Introduction of Authority Dynamics

The paper introduces the concept of Authority Dynamics, a framework for analyzing how authority behaves when automated and AI-driven systems traverse governance structures at machine speed. While prior work on governance focuses primarily on institutional structures and oversight mechanisms, this paper examines the operational behavior of authority during automated execution.

The Authority Stack Model

The paper proposes a two-dimensional analytical model—the Authority Stack—that represents the interaction between authority design layers and authority execution layers. This model provides a structured framework for analyzing how governance intent interacts with operational system environments and how automated systems traverse those layers during execution.

Identification of Dynamic Authority Phenomena

The paper identifies several dynamic behaviors that emerge when automated systems operate across governance architectures, including:

Authority layer compression

Basal primitive density and congestion

Authority propagation across execution layers

Elastic authority redistribution

Control path compression

Together, these phenomena describe how authority behaves in automated environments rather than simply where authority is formally located.

Governance Implications for Machine-Speed Systems

The paper demonstrates that governance mechanisms designed for sequential human oversight may be insufficient in machine-speed systems. Effective governance in automated environments increasingly requires control

mechanisms embedded within execution pathways and capable of operating at computational speed.

7. Authority Design Layers

Authority systems are typically designed as layered governance structures. These layers represent the conceptual levels through which intent, constraint, and decision authority are articulated within an organization or system. Higher layers tend to express goals, norms, and policy commitments, while lower layers translate those commitments into operational rules and technical mechanisms.

In practice, these governance layers are not always explicitly documented or formally separated. However, most complex systems implicitly rely on some form of layered design in order to translate institutional intent into executable system behavior. Without such layering, it becomes difficult to distinguish strategic direction from operational implementation or to identify where governance decisions originate.

For analytical purposes, this paper models authority design as a set of horizontal layers representing the structured placement of governance intent. These layers are not intended to represent specific departments or organizational roles. Rather, they represent functional categories through which authority is articulated and constrained.

A representative set of authority design layers can be described as follows:

Governance and Policy

High-level commitments that define institutional authority, legal constraints, and policy intent.

Strategic Objectives

Organizational priorities that guide decision-making and resource allocation.

Compliance and Risk

Mechanisms that ensure alignment with legal requirements, regulatory obligations, and risk management policies.

Ethical and Normative Constraints

Normative frameworks that guide acceptable system behavior beyond formal legal compliance.

Operational Standards

Procedures and operational guidelines that translate governance goals into repeatable practices.

Technical Controls

Technical mechanisms that enforce rules and constraints within systems, including configuration rules, security controls, and validation mechanisms.

Performance and Monitoring Metrics

Measurement and evaluation mechanisms that track system behavior, outcomes, and adherence to governance objectives.

These layers represent the designed structure of authority within a system. They capture how authority is intentionally distributed and constrained across governance mechanisms. In traditional governance models, decisions are expected to move sequentially across these layers: policy defines intent, operational standards interpret that intent, and technical systems execute the resulting actions.

However, this sequential model becomes increasingly strained in automated systems, where execution may traverse multiple governance layers simultaneously. To understand how authority behaves under those conditions, it is necessary to distinguish governance design layers from the layers through which authority is actually executed.

8. Authority Execution Layers

While authority design layers describe how governance intent is structured, system behavior occurs through a different set of layers: the operational environments in which decisions are executed and consequences appear. These layers represent the practical pathways through which authority manifests during system operation.

Execution layers are therefore vertical rather than horizontal. They describe the progression from user interaction and system computation through infrastructure, monitoring, and real-world outcomes. Authority that is designed in governance structures ultimately becomes visible only through activity within these execution environments.

For analytical clarity, this paper models authority execution across seven representative layers:

User Interface and Interaction

The interfaces through which users interact with systems, submit inputs, and receive outputs.

Model Inference and Decisioning

Computational processes that generate recommendations, predictions, or automated decisions.

Data Management and Processing

Systems responsible for storing, transforming, and delivering data used by models and applications.

Deployment and Infrastructure

The runtime environment in which software systems operate, including cloud platforms, orchestration systems, and compute resources.

System Monitoring and Auditing

Observability mechanisms that track system behavior, logs, and performance metrics.

Operational Feedback Loops

Processes through which system outcomes influence subsequent decisions, updates, or model retraining.

Real-World Outcomes and Impact

The layer where system behavior produces observable consequences in financial markets, physical systems, institutions, or social environments.

These execution layers represent the operational environment through which authority is enacted. Decisions occurring at higher design layers eventually manifest within these operational contexts. For example, a governance policy may ultimately influence how a model is deployed, how data is processed, or how a user receives an automated recommendation.

The separation between design layers and execution layers allows authority to be examined not only as a governance structure but also as an operational process. This distinction becomes particularly important when automated systems traverse governance constraints during execution rather than during explicit human decision-making.

9. The 7×7 Authority Stack

To analyze how governance design interacts with system execution, this paper introduces the Authority Stack, a two-dimensional model that combines authority design layers and authority execution layers into a unified analytical framework.

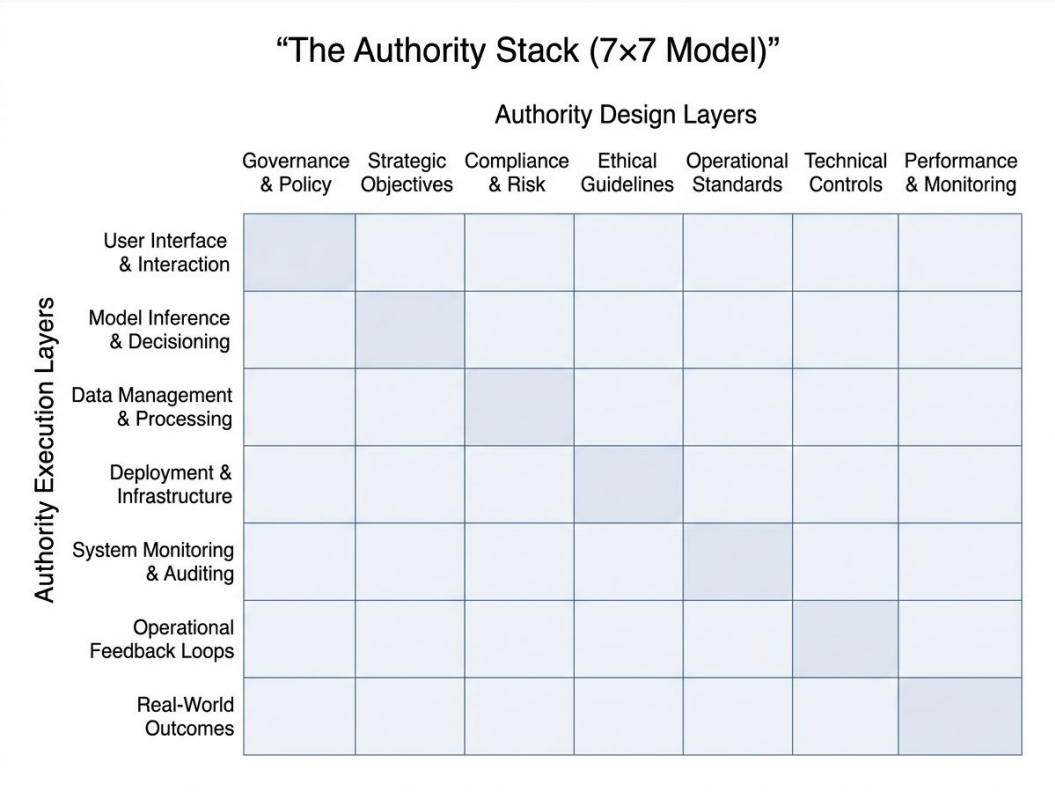


Figure 1. The Authority Stack (7×7 Model)

Let the authority stack be represented as:

$$S(i, j)$$

where:

i represents the design layer

j represents the execution layer

Each position in the stack represents the interaction between a governance design constraint and an operational execution environment.

In the model presented here:

$$i \in \{1 \dots 7\}$$
$$j \in \{1 \dots 7\}$$

resulting in a 7×7 authority stack.

Conceptually, this structure forms a grid in which:

horizontal movement represents movement across governance intent

vertical movement represents movement through operational execution

individual cells represent points where governance decisions interact with system activity

For example, a policy constraint applied to model deployment may occupy the intersection between a governance layer such as compliance and risk and an execution layer such as deployment and infrastructure. Similarly, a performance metric used to evaluate system behavior may connect monitoring mechanisms with governance performance objectives.

This structure allows authority to be analyzed as a system of interacting governance and execution environments rather than as a single hierarchical chain of command.

More importantly, the authority stack provides a way to analyze how automated systems traverse governance structures. Human decision-making often moves across governance layers sequentially: policy is interpreted, compliance checks are performed, operational standards are applied, and technical systems are executed. Automated systems may instead traverse multiple cells of the authority stack simultaneously during a single computational event.

When such traversal occurs, governance layers that were designed to remain distinct may become compressed into a single execution pathway. This compression reveals dynamic properties of authority systems that are difficult to observe in slower, human-mediated processes.

The next sections examine how automated systems and AI models move through this authority stack and how that movement produces observable dynamics such as governance layer compression, primitive density clustering, and control path shortening.

10. AI Traversal of Governance Layers

Traditional governance models assume that authority flows through systems in a sequential manner. Decisions are expected to move through governance layers step by step: policy defines intent, strategy interprets that intent, compliance mechanisms verify constraints, operational procedures guide execution, and technical systems implement the resulting actions. Oversight mechanisms then evaluate outcomes and feed information back into the governance process.

This sequential model reflects the pace of human decision-making. Human institutions typically operate through deliberation, review processes, approval chains, and procedural checkpoints. Each governance layer therefore acts as a stage in a decision pathway.

Automated systems and AI models alter this behavior fundamentally. In automated execution environments, decisions are often produced within milliseconds or microseconds. A single system event—such as a model inference, automated trade, deployment action, or recommendation—may simultaneously instantiate assumptions from multiple governance layers. Policy constraints, risk limits, operational rules, technical controls, and performance objectives may all be encoded within the system that produces the decision.

As a result, automated execution does not move sequentially through governance layers in the same way that human processes do. Instead, automated systems often traverse multiple governance layers within the same computational event.

“AI Traversal and Authority Layer Compression”

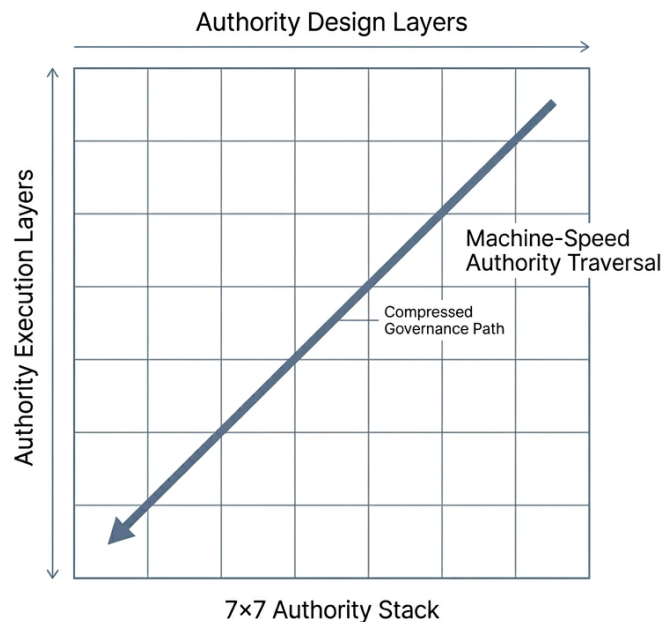


Figure 2. AI Traversal and Authority Layer Compression

This phenomenon can be described as machine-speed authority traversal.

Machine-speed authority refers to the condition in which authority embedded across governance layers is activated and executed at computational speed rather than through sequential human processes. In such systems, the governance structure still exists, but it is encountered simultaneously rather than stepwise during execution.

Within the Authority Stack model introduced in the previous section, this behavior can be understood as a traversal path across multiple cells of the stack during a single execution event. Rather than moving horizontally across governance layers before descending into execution, automated systems may intersect multiple design and execution layers at once. A model inference may draw on data governance constraints, risk controls, operational objectives, and technical

deployment rules simultaneously while producing an output that affects real-world outcomes.

This traversal behavior reveals a critical property of automated systems: governance layers are not removed, but they are encountered in compressed form during execution. The layers that were designed to remain analytically distinct may therefore appear to collapse into a single operational pathway.

Understanding this traversal pattern is essential for analyzing authority in automated environments. If governance mechanisms are designed under the assumption of sequential review but execution occurs through simultaneous traversal, oversight mechanisms may fail to operate as intended. The governance architecture may remain intact in principle, but its practical behavior changes when authority moves through it at machine speed.

The next section examines the primary dynamic consequence of this traversal behavior: authority layer compression, in which governance layers designed to operate sequentially become compressed into single execution pathways during automated decision processes.

11. Authority Layer Compression

When automated systems traverse governance layers at machine speed, the distinction between layers designed for sequential decision-making can become compressed during execution. This phenomenon can be described as authority layer compression.

Authority layer compression occurs when governance layers that are structurally distinct within the authority architecture are encountered simultaneously during automated execution. In such cases, the layered design of authority remains present in the system, but the execution pathway passes through those layers within a single operational event rather than across multiple decision stages.

In traditional governance processes, layers such as policy, compliance, operational procedures, and technical implementation are typically separated by time and human review. Decisions move through these layers sequentially, allowing oversight mechanisms to intervene at intermediate stages. Each layer therefore acts as a checkpoint that interprets and constrains authority before execution occurs.

Automated systems reduce or eliminate the temporal separation between these layers. When governance rules are encoded within software systems, model parameters, deployment pipelines, or automated decision engines, those constraints may be applied simultaneously when a decision is executed. Rather than passing through a series of intermediate approvals, the decision path may incorporate multiple governance constraints at once.

Within the Authority Stack model, this effect can be represented as the traversal of several cells of the stack during a single execution event. Instead of moving stepwise across design layers before descending into execution layers, automated processes may intersect multiple design and execution layers simultaneously. A single system event may therefore instantiate governance intent, compliance limits, operational rules, and technical controls at the same moment.

This compression does not imply that governance layers disappear or lose significance. On the contrary, the underlying architecture continues to shape system behavior. However, the temporal spacing between governance layers collapses under automated execution, producing a compressed authority pathway.

Authority layer compression has several implications for governance. First, it reduces the opportunity for intermediate oversight that relies on sequential review. If governance checkpoints operate more slowly than automated execution, they may fail to intervene before decisions propagate through the system. Second, compression increases the complexity of analyzing authority pathways, because multiple governance constraints may influence a decision simultaneously rather than sequentially. Third, compressed pathways can amplify the speed at which decisions propagate through execution layers and into real-world outcomes.

Understanding authority layer compression is therefore essential for analyzing governance in automated environments. It explains why systems that appear well-governed at the architectural level may still produce unexpected or uncontrolled behavior when authority moves through them at computational speed.

While layer compression describes how governance layers collapse temporally during execution, additional dynamics emerge within the lower regions of the authority architecture where operational primitives interact most intensely. The next section examines this phenomenon by introducing the concept of basal

primitive density and the resulting conditions of basal congestion within the authority stack.

12. Basal Primitive Density and Congestion

While authority layer compression describes how governance layers collapse temporally during automated execution, additional dynamics emerge within the lower regions of the authority architecture. These dynamics arise from the interaction of fundamental governance primitives at the point where abstract governance intent is translated into operational execution.

In the Authority Architecture framework introduced in Paper 1, authority is constructed from a set of primitives—foundational elements such as policy, identity, money, models, contracts, or physical control. These primitives interact to produce governance structures and operational behaviors. While such primitives may appear throughout the authority architecture, their interactions tend to concentrate near the lowest design layers where operational rules and technical controls are implemented.

This concentration produces a phenomenon that can be described as basal primitive density.

Basal primitive density refers to the increasing concentration of primitive interactions within the lowest layers of the authority design structure. These layers sit closest to operational execution, where governance constraints must ultimately be expressed in enforceable mechanisms such as technical controls, automated rules, or system configurations. As a result, multiple primitives frequently intersect at this level in order to produce executable decisions.

For example, a single automated decision within a financial system may involve interactions between identity primitives (who is authorized to act), monetary primitives (what value is being transferred), contractual primitives (what obligations govern the transaction), and policy primitives (what regulatory or organizational constraints apply). Although these primitives originate in different parts of the governance architecture, their effects often converge near the point where execution occurs.

As automation increases, the rate of these interactions grows significantly. Automated systems continuously generate decisions, perform evaluations, and propagate outcomes across execution layers. Each of these events may involve

multiple primitive interactions, and when many such events occur concurrently the density of primitive interactions within the basal layer increases.

When primitive density becomes sufficiently high, the lowest design layer may exhibit basal congestion.

Basal congestion occurs when the concentration of primitive interactions within the basal design layer becomes difficult to analytically separate. In such conditions, individual primitives may still exist and operate correctly, but their interactions become tightly coupled. The observable behavior of the system therefore reflects clusters of interacting primitives rather than clearly distinguishable individual mechanisms.

This phenomenon does not imply that primitives themselves collapse or lose their identity. Rather, it reflects a limitation in representational clarity when many primitive interactions occur simultaneously within a narrow architectural region. Under heavy automation or AI-driven decision activity, the basal layer of the authority architecture may therefore appear densely clustered with overlapping interactions.

Basal Primitive Density and Congestion

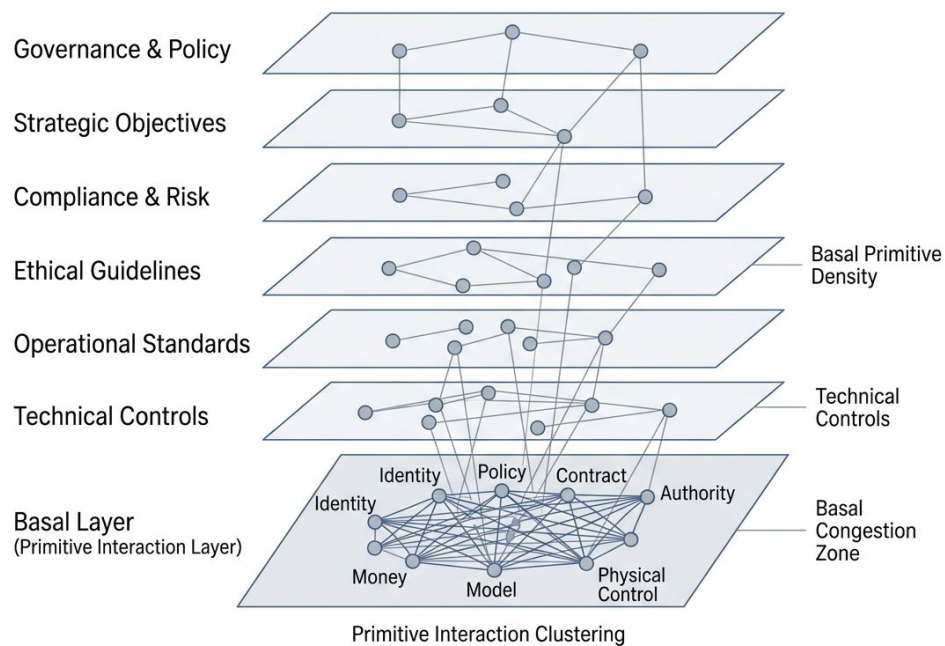


Figure 3. Basal Primitive Density and Congestion

Understanding basal primitive density and congestion is important for analyzing authority behavior in automated systems. It explains why the lowest governance layers—those closest to operational execution—often exhibit the most complex interaction patterns. It also provides a basis for explaining why governance models that treat primitives as cleanly separable elements may struggle to represent the dynamics of highly automated systems.

When primitive density exceeds the level at which interactions can be clearly distinguished, the architectural representation of the basal layer may need to be expanded in order to preserve analytical clarity. The next section introduces the concept of foundational decomposition, a representation rule that allows the basal layer to be expanded when primitive separability falls below a defined threshold.

13. Foundational Decomposition

As primitive interactions accumulate within the basal region of the authority architecture, the density of those interactions can reach a point where analytical separation becomes difficult. Although the primitives themselves remain distinct in structure, the observable behavior of the system may appear as tightly coupled clusters of interacting mechanisms. In such conditions, representing the basal layer as a single architectural layer may no longer provide sufficient resolution to clearly distinguish the underlying primitives and their interactions.

To preserve analytical clarity under these conditions, this paper introduces the concept of foundational decomposition.

Foundational decomposition is a representational technique used when the density of primitive interactions within the basal layer exceeds the threshold at which those primitives can be clearly distinguished. Rather than altering the underlying authority architecture, foundational decomposition expands the representational resolution of the basal layer so that individual primitive interactions can be analyzed separately.

This process can be understood as a change in analytical scale. At low levels of primitive interaction, the basal layer may be modeled as a single layer within the authority stack. However, when automated systems produce a high frequency of primitive interactions—such as in AI-driven decision environments or large-scale automated infrastructures—the number of interactions occurring within that layer

may increase significantly. In such cases, maintaining a single aggregated layer may obscure important distinctions between primitives and their effects.

Formally, this condition can be described in terms of primitive separability. Let σ represent the separability of primitives within a region of the authority architecture. Separability refers to the ability to analytically distinguish individual primitives and their interactions within that region.

When the separability of primitives within the basal layer falls below a defined threshold, the layer may be decomposed into additional sublayers for analytical purposes. This rule can be expressed as:

If the separability of the basal layer is less than a defined threshold, the basal layer should be decomposed to preserve primitive distinction.

Importantly, this rule applies only to the representation of the architecture, not to the architecture itself. Foundational decomposition does not introduce new primitives or alter the underlying authority structure. Instead, it increases the resolution of the model used to describe that structure.

This distinction is essential. The purpose of decomposition is not to change the authority system, but to maintain clarity in its representation when interaction density becomes high. In effect, foundational decomposition allows analysts to zoom in on the basal layer of the authority architecture in order to observe primitive interactions that would otherwise appear compressed.

Once decomposition occurs, it becomes possible to examine how primitive interactions propagate across adjacent layers of the authority stack. These interactions may influence not only the basal layer itself but also the behavior of neighboring layers within the architecture. The next section therefore examines how authority effects propagate through the stack and how systems respond to congestion through processes of elastic authority redistribution.

14. Authority Propagation and Elastic Redistribution

When automated systems traverse the authority stack at machine speed, decisions and their consequences propagate rapidly across both design and execution layers. A single execution event may trigger effects that move vertically through operational layers while simultaneously interacting with multiple governance constraints embedded in the design layers. These interactions

create pathways through which authority influences system behavior beyond the initial decision point.

These pathways can be described as authority propagation paths.

An authority propagation path is the sequence of interactions through which a decision or constraint moves across the authority stack. Propagation may occur vertically through execution layers as decisions move from model inference to infrastructure, monitoring systems, and ultimately real-world outcomes. It may also occur horizontally when governance constraints embedded in different design layers simultaneously shape a single execution event.

In automated systems, propagation occurs at computational speed. Decisions generated within a model or automated control system may immediately affect deployment infrastructure, system monitoring, operational feedback loops, and real-world outcomes. Because multiple layers are involved simultaneously, authority effects can move rapidly through the stack before traditional oversight mechanisms have time to intervene.

When high densities of primitive interaction occur within the basal layer, these propagation paths may become constrained. The concentration of interactions in the basal region can create local congestion, where multiple primitives compete to shape execution behavior within a narrow architectural region. This congestion does not necessarily cause system failure, but it alters how authority pressures move across the stack.

In response to such conditions, authority effects may redistribute across adjacent layers. This behavior can be described as elastic authority redistribution.

Elastic authority redistribution refers to the tendency of authority pressures to shift across neighboring layers of the authority stack when interaction density or congestion increases in a particular region. Rather than collapsing the governance structure, the system redistributes the influence of constraints and controls across nearby layers in order to maintain operational behavior.

For example, if a high density of primitive interactions occurs within technical control mechanisms, additional constraints may emerge through monitoring systems, operational feedback loops, or higher-level governance adjustments. Similarly, when operational constraints become difficult to enforce within one layer of execution, enforcement mechanisms may shift to adjacent layers such as infrastructure controls or policy-driven monitoring.

This elastic behavior reflects an important property of complex authority systems: governance structures rarely fail through a single point of collapse. Instead, pressure within one region of the architecture often produces adjustments elsewhere in the system. Authority therefore behaves less like a rigid hierarchy and more like a distributed control field whose effects propagate and redistribute across layers.

Understanding propagation and redistribution dynamics is important for explaining how automated systems maintain operational stability even when governance layers are compressed. These dynamics also reveal why governance mechanisms designed for slower decision processes may fail to control automated systems effectively. If control mechanisms operate outside the primary propagation paths of automated execution, they may be bypassed by machine-speed decision pathways.

This leads to the final dynamic phenomenon examined in this paper: control path compression, in which oversight mechanisms must operate within the same execution layers and time scales as the automated systems they govern.

15. Control Path Compression and Governance Response

The dynamics described in the preceding sections—authority traversal, layer compression, basal primitive density, and elastic redistribution—have a common implication: automated systems propagate authority effects at speeds and across layers that often exceed the operational capacity of traditional governance mechanisms. When execution occurs at machine speed, oversight structures that depend on sequential human review may be unable to intervene before decisions have already propagated through the system.

This condition produces a phenomenon that can be described as control path compression.

Execution Paths vs Control Path Compression

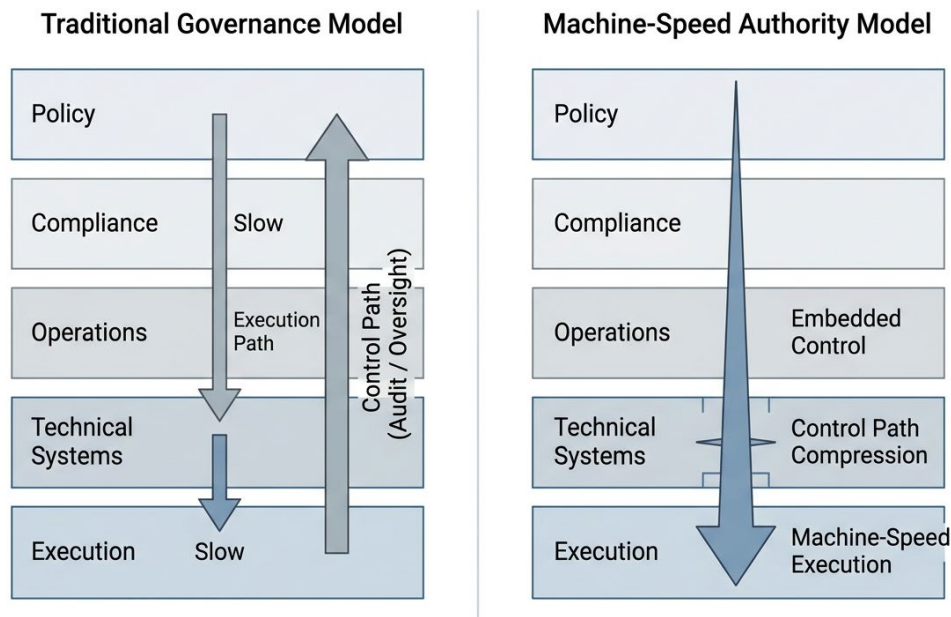


Figure 4. Execution Paths vs Control Path Compression

Control path compression occurs when effective governance and oversight must operate within the same execution layers and temporal scales as automated decision processes. In traditional governance models, control paths are typically separated from execution paths. Decisions are made within operational systems, while oversight occurs through later auditing, reporting, or supervisory review. The control path therefore operates at a slower pace than the execution path.

Automated systems reduce or eliminate this separation. When decisions propagate through the authority stack in milliseconds, oversight mechanisms that rely on delayed analysis or retrospective review cannot prevent undesired outcomes. They can only observe effects after they have already occurred. In such environments, governance mechanisms that operate outside the execution pathway may be structurally incapable of controlling system behavior.

Within the Authority Stack model, this distinction can be expressed in terms of execution paths and control paths. Execution paths represent the operational routes through which automated decisions propagate across execution layers.

Control paths represent the mechanisms through which governance constraints monitor, constrain, or modify those decisions.

When control paths operate more slowly than execution paths, the effective authority of governance mechanisms diminishes. Authority may still exist architecturally within governance layers, but its ability to influence system behavior becomes limited by the speed and location of the mechanisms that enforce it.

Control path compression therefore describes the process by which governance mechanisms must move closer to the point of execution in order to remain effective. Oversight mechanisms may need to operate within the same infrastructure, data processing environments, or decision engines that produce automated outputs. In some cases, governance rules must be encoded directly into technical systems so that constraints are applied at the moment of execution rather than during later review.

This shift does not eliminate the role of higher-level governance structures such as policy or strategic oversight. Instead, it requires those structures to be translated into mechanisms capable of operating within machine-speed execution environments. Governance intent remains defined at higher layers of the authority architecture, but its enforcement must occur within the operational layers where automated decisions are produced.

Understanding control path compression is essential for explaining why many traditional governance mechanisms struggle to regulate highly automated systems. It also clarifies the conditions under which governance can remain effective: oversight must be capable of operating within the same authority propagation paths and time scales as the automated processes it seeks to govern.

The final section of this paper examines the broader implications of these dynamics for the governance of AI-driven and automated systems.

16. Implications for AI Governance and Automated Systems

The dynamics described in this paper have direct implications for the governance of automated and AI-driven systems. Many contemporary governance frameworks assume that authority operates through sequential institutional

processes: policies are defined, compliance mechanisms interpret those policies, operational procedures guide implementation, and oversight mechanisms evaluate outcomes after execution. These models implicitly assume that governance layers remain temporally separated and that human decision-makers can intervene between stages of system behavior.

The analysis presented in this paper suggests that this assumption does not hold in highly automated environments. When decisions are generated and executed at machine speed, authority embedded across governance layers is encountered simultaneously during execution. As a result, governance mechanisms designed for sequential review may fail to influence system behavior before decisions propagate across operational layers and produce real-world effects.

This observation has several implications for the design and evaluation of governance systems.

First, governance frameworks must account for machine-speed authority traversal. Policies, compliance constraints, and ethical guidelines cannot rely solely on human interpretation during operational decision-making. Instead, these governance commitments must be translated into mechanisms that can operate within the same computational environments where automated decisions are produced. Without such translation, governance intent may remain formally defined but operationally disconnected from system execution.

Second, governance systems must recognize the effects of authority layer compression. When multiple governance constraints are encountered simultaneously during automated execution, oversight mechanisms must be capable of evaluating decisions within the compressed pathway rather than relying on sequential checkpoints. This may require integrated governance mechanisms that operate directly within system infrastructure, data pipelines, or model execution environments.

Third, governance models must account for basal primitive density within operational layers. Automated systems frequently concentrate interactions among identity, monetary, contractual, and policy primitives near the points where decisions are executed. As automation scales, the density of these interactions increases. Governance mechanisms that attempt to evaluate primitives independently may struggle to represent the tightly coupled interaction clusters that occur in practice. Analytical tools and system architectures may therefore need to account for interaction density and clustering within operational layers.

Fourth, the presence of elastic authority redistribution suggests that governance pressures often shift across adjacent layers rather than collapsing at a single point. When enforcement mechanisms fail or become congested within one region of the authority stack, constraints may emerge through monitoring systems, infrastructure controls, or operational feedback loops. Effective governance therefore requires visibility across the entire authority stack rather than focusing exclusively on a single governance layer.

Finally, the phenomenon of control path compression implies that oversight mechanisms must increasingly operate within the same environments and time scales as automated execution. Governance mechanisms that rely solely on retrospective auditing or external supervision may be unable to influence system behavior before outcomes occur. Instead, effective oversight may require mechanisms that are embedded within execution environments themselves.

These implications do not suggest that traditional governance structures are obsolete. Policies, legal frameworks, and institutional oversight remain essential for defining the boundaries of acceptable system behavior. However, the effectiveness of those structures depends on their ability to translate governance intent into mechanisms capable of operating within automated execution environments.

Understanding the dynamics of authority traversal, compression, and propagation therefore provides a foundation for designing governance systems that remain effective under conditions of large-scale automation and AI-driven decision-making.

17. Conclusion

This paper has examined how authority behaves in automated and AI-driven systems. While authority is typically designed as a layered governance structure, automated execution changes the way those layers are encountered during system operation. Instead of moving sequentially through governance stages, automated systems frequently traverse multiple layers simultaneously during a single execution event. This behavior reveals dynamic properties of authority that are difficult to observe in slower, human-mediated processes.

To analyze these dynamics, the paper introduced the Authority Stack, a two-dimensional model that combines authority design layers with authority execution layers. This framework provides a way to examine how governance intent

interacts with operational environments and how automated systems move through those layers during execution. Within this structure, automated decision processes can be understood as traversal paths that intersect multiple governance constraints while propagating through execution environments.

Several dynamic phenomena emerge from this traversal behavior. Authority layer compression occurs when governance layers designed for sequential interpretation are encountered simultaneously during automated execution. Basal primitive density describes the increasing concentration of primitive interactions within the lowest design layers where governance intent is translated into operational rules. When interaction density becomes sufficiently high, basal congestion may occur, requiring representational expansion through foundational decomposition in order to preserve analytical clarity.

The paper also examined how authority effects propagate across the authority stack. Automated decisions may move rapidly across execution layers while interacting with multiple governance constraints. Under conditions of high interaction density, authority pressures may redistribute across adjacent layers through processes of elastic authority redistribution. These dynamics help explain why governance structures rarely fail through a single point of collapse and instead exhibit distributed adjustment across multiple layers of the system.

Finally, the paper introduced the concept of control path compression, which describes the need for governance mechanisms to operate within the same layers and time scales as automated execution. Oversight mechanisms that operate more slowly than the systems they govern may observe outcomes after the fact but may be unable to influence behavior before those outcomes occur. Effective governance in automated environments therefore requires mechanisms capable of operating within machine-speed execution pathways.

Taken together, these dynamics suggest that automation does not eliminate authority architecture but instead reveals its operational behavior. Authority systems remain structured through governance layers, but their behavior changes when automated systems traverse those layers at computational speed. Understanding these dynamics is necessary for analyzing modern authority systems and for explaining why governance mechanisms designed for slower decision processes often struggle to regulate automated environments.

Authority Dynamics in Automated and AI-Driven Systems

Machine-speed traversal of governance layers produces dynamic authority behaviors.

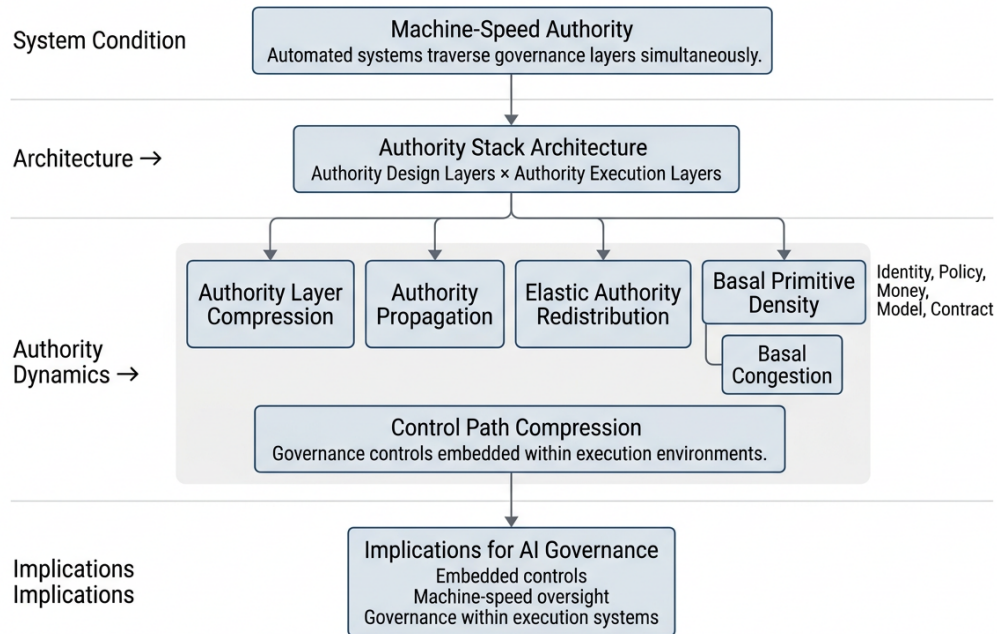
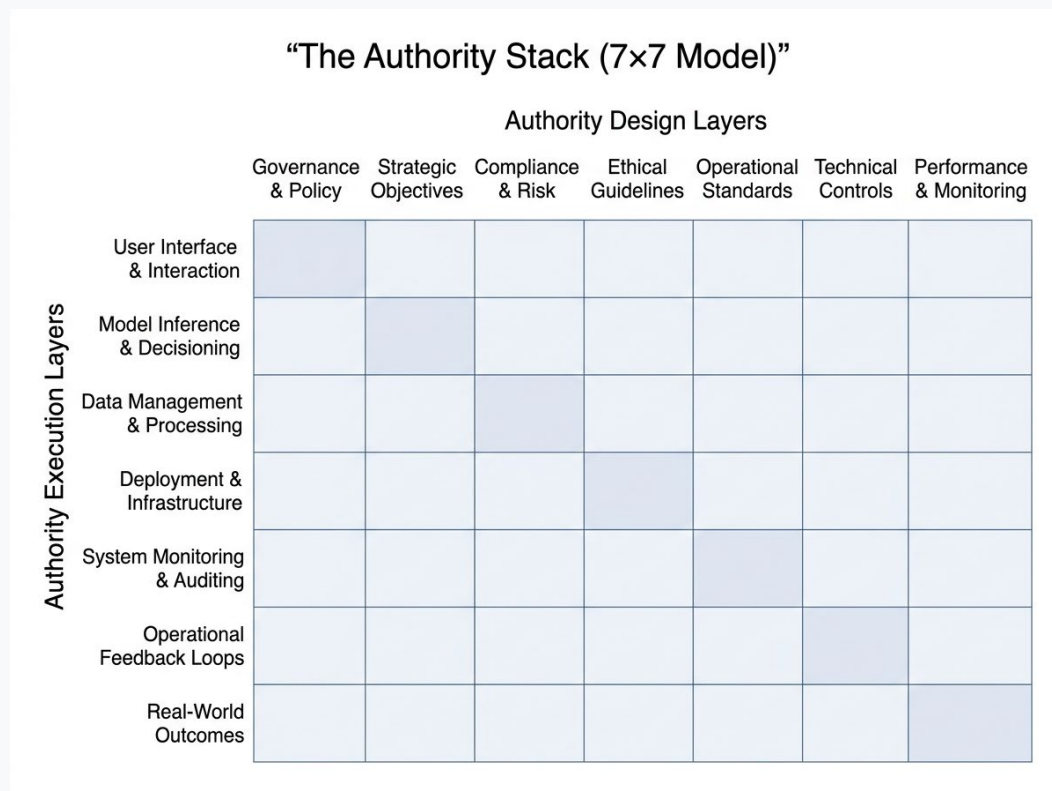


Figure 5. Authority Dynamics in Automated and AI-Driven Systems

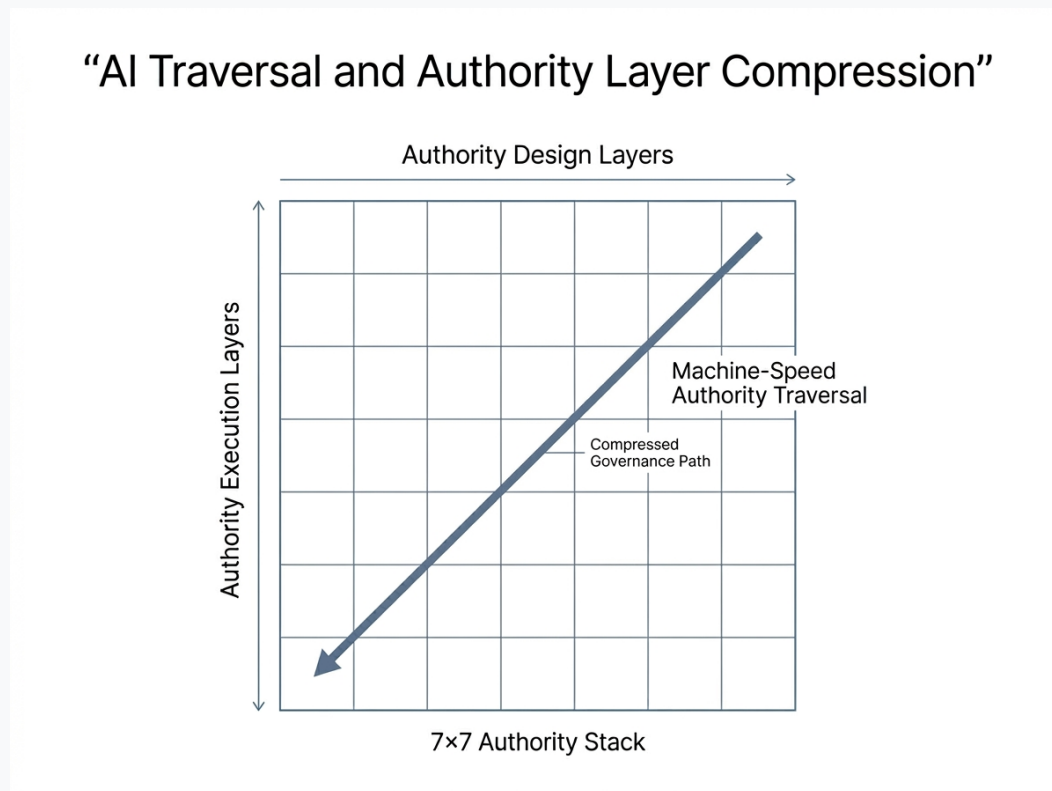
Figures

Figure 1. The Authority Stack (7×7 Model)



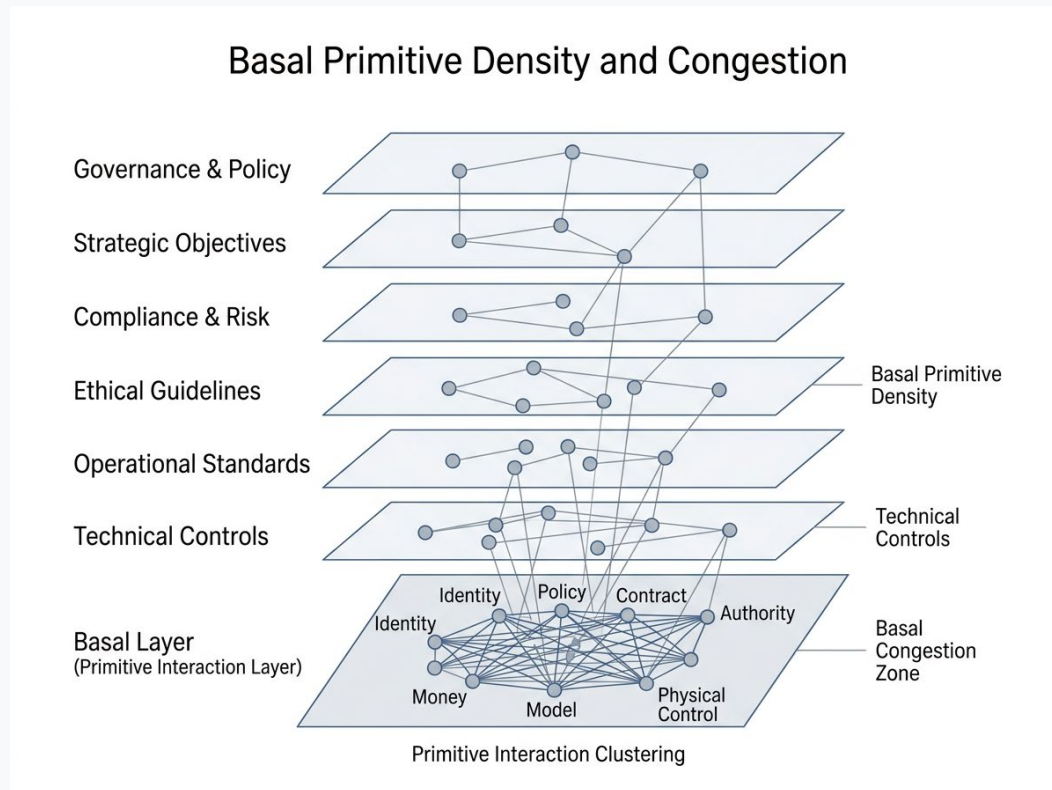
The Authority Stack represents the interaction between authority design layers and authority execution layers. Each cell represents a point where governance constraints interact with operational system behavior.

Figure 2. AI Traversal and Authority Layer Compression



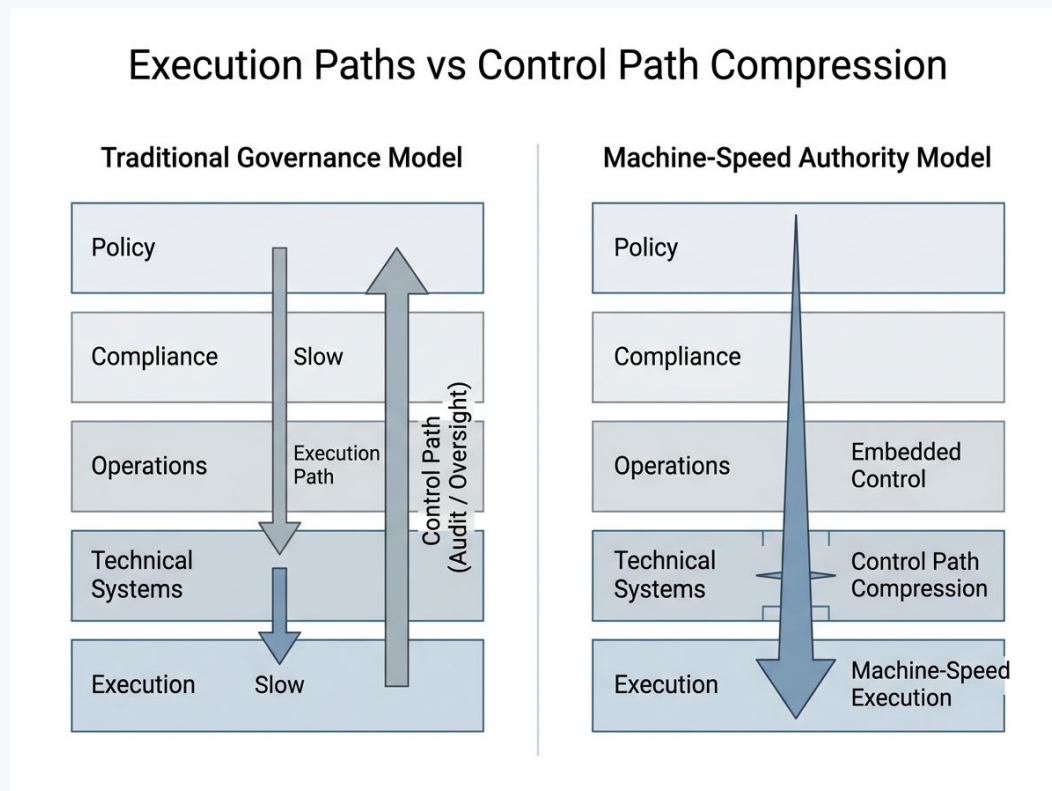
Automated systems may traverse multiple governance layers simultaneously during execution, compressing authority pathways that were originally designed to operate sequentially.

Figure 3. Basal Primitive Density and Congestion in the Authority Architecture



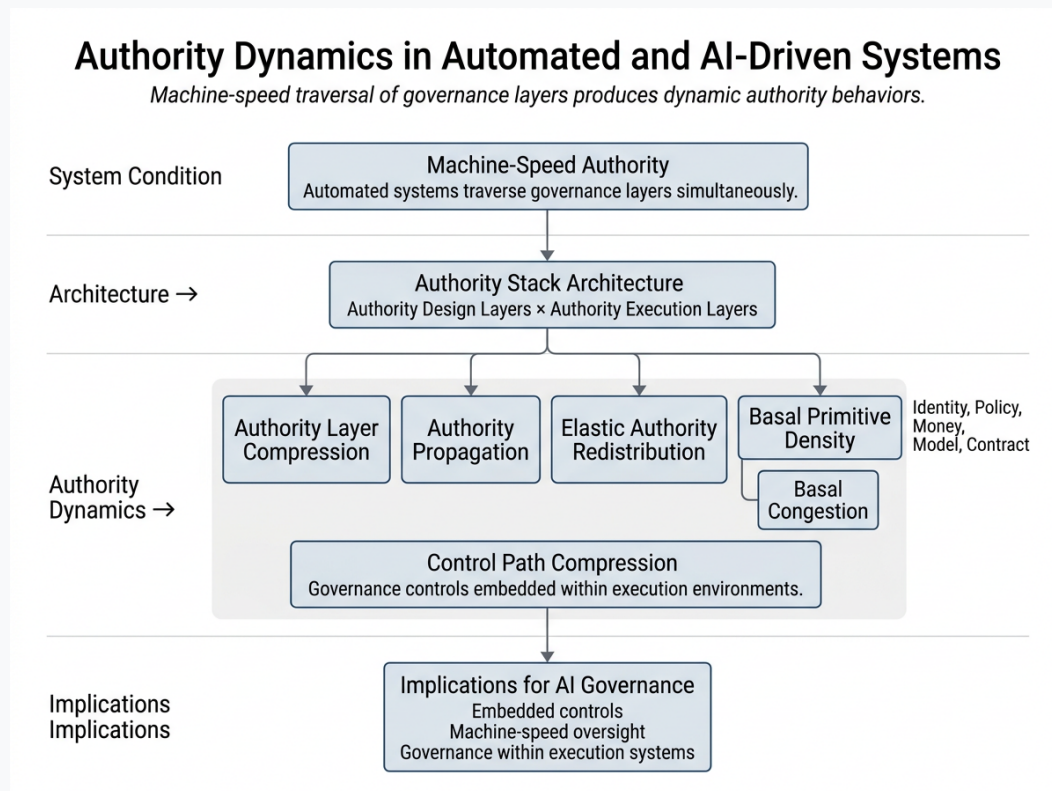
Primitive interactions cluster in the lowest layers of the authority architecture where governance intent is translated into operational rules. Under high automation, primitive density increases and may produce basal congestion.

Figure 4. Execution Paths vs Control Compression



Traditional governance relies on oversight paths that operate outside the execution path. In automated systems, effective control mechanisms must operate within the execution pathway at machine speed.

Figure 5. Authority Dynamics in Automated and AI-Driven Systems.



Machine-speed traversal of governance layers exposes dynamic behaviors within authority architectures. Automated execution compresses governance layers, increases primitive density within basal layers, and propagates authority effects across the stack. These dynamics require governance mechanisms embedded within execution environments through control path compression and machine-speed oversight.

References

- Lessig, L. (1999).
Code and Other Laws of Cyberspace.
Basic Books.
- Bostrom, N. (2014).
Superintelligence: Paths, Dangers, Strategies.
Oxford University Press.
- Varian, H. (2019).
Artificial intelligence, economics, and industrial organization.
Innovation Policy and the Economy, 19, 1–40.
- Rahwan, I. (2018).
Society-in-the-loop: Programming the algorithmic social contract.
Ethics and Information Technology, 20(1), 5–14.
- Zuboff, S. (2019).
The Age of Surveillance Capitalism.
PublicAffairs.
- Kroll, J., Huey, J., Barocas, S., et al. (2017).
Accountable algorithms.
University of Pennsylvania Law Review, 165(3), 633–705.
- Diakopoulos, N. (2016).
Accountability in algorithmic decision making.
Communications of the ACM, 59(2), 56–62.
- Ostrom, E. (2005).
Understanding Institutional Diversity.
Princeton University Press.