Computation After Storage:
Toward an Entropic Theory of Semantic Infrastructure

Flyxion

December 28, 2025

Abstract

This essay argues that storage-centric models of computation—organized around files, snapshots, and persistent state—are artifacts of reversible abstraction rather than reflections of computational reality. As computation has become distributed, collaborative, and semantically heterogeneous, these models no longer adequately describe how meaning is produced, maintained, or reconciled.

We propose an alternative perspective in which computation is understood as an irreversible, constraint-preserving semantic process operating over local contexts. From this standpoint, merge is not a lossless operation but an entropy-increasing physical event; infrastructure is not a repository but a field of constraints; and state is not primary but derived from event-historical processes. The resulting framework reframes computation as ongoing maintenance of coherence rather than execution over stored artifacts.

1 Introduction: The End of Storage as a Primitive

For much of the history of computing, storage has occupied a privileged conceptual position. Files, records, memory locations, and persistent objects have been treated not merely as implementation details, but as the fundamental units of computational meaning. Programs were understood as operating over stored state; correctness was defined relative to the contents of memory; coordination between agents was mediated through shared artifacts whose persistence was taken for granted.

This orientation was historically justified. Early computational systems were physically localized, temporally discrete, and materially constrained. Storage devices were scarce, expensive, and slow, and the act of persisting information was itself a dominant computational cost. Under these conditions, it was natural to treat stored state as the primary carrier of meaning and to regard computation as a sequence of transformations applied to that state.

However, the material and social conditions under which computation now occurs have changed decisively. Contemporary computation is distributed across networks, replicated across administrative domains, and entangled with human interpretation at every scale. The same nominal artifact may be read, modified, interpreted, and recombined by agents operating under divergent assumptions, incentives, and constraint regimes. In such environments, persistence alone no longer guarantees identity, and access to a stored representation no longer guarantees shared meaning.

Despite this shift, storage-centric metaphors continue to dominate both technical infrastructure and theoretical discourse. Files are still treated as authoritative sources of truth; version histories are assumed to encode objective pasts; and state snapshots are used as if they were neutral cross-sections of reality. These metaphors persist not because they remain adequate, but because they are deeply entrenched in institutional practice and cognitive habit.

The central claim of this essay is that storage is no longer a viable primitive for understanding computation. What persists in modern systems is not state, but constraint. Computation is best understood not as the manipulation of stored objects, but as an irreversible process of semantic transformation constrained by physical, logical, and social limits. Meaning is not retrieved from storage; it is actively maintained against entropy.

2 Storage Is a Reversible Fiction

The concept of storage rests on a constellation of assumptions that are rarely made explicit. First, storage presupposes stable identity: that an object written at one time can be read at another and remain *the same object* in all relevant respects. Second, it presupposes reversible access: that past states can be recovered without cost or distortion. Third, it presupposes negligible entropy production: that persistence itself does not meaningfully alter the system in which it occurs.

These assumptions are convenient, but they are false.

In physically realized systems, no stored representation is isolated from its environment. Bits decay, formats evolve, dependencies change, and interpretation drifts. Even when the raw symbols remain intact, the semantic context required to interpret them does not. A configuration file written under one software ecology may become ambiguous or meaningless under another; a data record preserved verbatim may encode assumptions that are no longer valid; a legal or technical document may remain syntactically unchanged while becoming semantically unstable.

Distributed systems exacerbate this instability. Replication introduces divergence; reconciliation introduces loss. When multiple agents independently modify replicas of a stored artifact, there is no privileged mechanism by which the resulting histories can be recombined without discarding information or introducing external judgment. The very act of attempting to restore coherence exposes the fiction of reversibility on which storage relies.

Storage therefore functions not as a physical primitive, but as an epistemic convenience: a simplifying abstraction that suppresses the costs of history, interpretation, and reconciliation. It allows systems to behave *as if* meaning were persistent, reversible, and context-free, even when it is not.

Remark 1. *The persistence of storage metaphors reflects institutional inertia rather than ontological necessity. Storage survives as a scaffold for coordination long after the conditions that justified its centrality have disappeared.*

3 Computation as Irreversible Process

The claim that computation is an irreversible process is not merely a metaphor borrowed from thermodynamics, but a statement about the physical conditions under which computation is realized. To compute is to effect a transformation in a physical substrate, and all such transformations occur in time, consume energy, and interact with an environment that is not fully observable or controllable.

Classical models of computation obscure this fact by treating computation as an abstract relation between inputs and outputs. In such models, a computation is fully characterized by a function, and functions may be inverted, composed, or rewound without consequence. Time plays no essential role; history is incidental. This abstraction has proven extraordinarily useful, but it achieves its power by suppressing precisely those features that dominate real computational systems: noise, dissipation, and context-dependence.

In physically instantiated systems, computation leaves traces. Intermediate states are written into memory, signals propagate through space, and interactions with uncontrolled degrees of freedom accumulate. Even when a computation is logically reversible in principle, its physical realization is not. Reversing a computation would require reconstructing not only the symbolic state of the system, but also the precise microstate of its environment, a task that is physically impossible beyond trivial scales.

Irreversibility therefore enters computation not as a contingent flaw of implementation, but as a structural feature of embodied processes. The past of a computation cannot be undone; it can only be summarized, forgotten, or reinterpreted. Any framework that treats rollback, undo, or reversal as primitive confuses representational convenience with physical reality.

This distinction becomes unavoidable once computation is distributed across multiple agents. When independent processes evolve under partial information and later interact, there is no canonical joint history. Reconciliation requires choice, and choice requires loss. The arrow of time asserts itself not only in energy dissipation, but in the collapse of alternative semantic trajectories.

Computation, in this sense, is better understood as a form of irreversible history-writing constrained by admissibility rather than as execution over a persistent state space.

4 Thermodynamic Foundations of Computation

The irreversibility of computation admits a precise thermodynamic formulation. Any physical realization of computation must obey the laws of statistical mechanics, and in particular the second law of thermodynamics. This places non-negotiable lower bounds on the energetic and entropic costs of information processing.

A central result in this domain is Landauer's principle, which states that the erasure of one bit of information requires the dissipation of at least $k_B T \ln 2$ of heat into the environment. While Landauer's principle is often invoked narrowly in discussions of memory erasure, its implications are far more general. Any operation that reduces the distinguishability of physical states is an erasure in this sense, regardless of whether it is labeled as such at the logical level.

To formalize this insight in a semantic setting, we introduce the notion of computational entropy associated with a transformation between admissible states.

Definition 1 (Computational Entropy)**.** *Let s_0 and s_1 be admissible semantic states realized by ensembles of physical microstates with cardinalities $\Omega(s_0)$ and $\Omega(s_1)$, respectively. The computational entropy produced by a transformation $s_0 \to s_1$ is defined as*

$$\Delta S(s_0 \to s_1) \;=\; k_B \ln \Omega(s_1) - k_B \ln \Omega(s_0) \;+\; S_{\text{diss}},$$

where S_{diss} accounts for entropy dissipated into unobserved degrees of freedom during the transformation.

This definition separates two sources of entropy production. The first arises from changes in the cardinality of the macrostate itself: moving from a more constrained to a less constrained semantic description necessarily increases entropy. The second arises from dissipation required to enforce constraints, maintain correlations, and suppress noise.

Crucially, constraint preservation is not entropy-neutral. Maintaining a semantic invariant against thermal and stochastic perturbations requires active error correction, feedback, and control, all of which incur energetic cost. The illusion that constraints are free arises only when their maintenance is outsourced to infrastructure and treated as background.

We can now state a general irreversibility result.

[Irreversibility of Constraint-Preserving Computation] Let $T : S \to S$ be a physically realized computational transformation that is many-to-one on semantic states, preserves a nontrivial set of semantic constraints, and operates in finite time using finite energy. Then the computational entropy production $\Delta S(T)$ is strictly positive.

Proof. Because T is many-to-one, there exist distinct semantic states s and s' such that $T(s) = T(s')$. Any physical realization of T must therefore map distinct ensembles of microstates into a common ensemble, eliminating distinguishability. By Landauer's principle, this requires dissipation of entropy at least proportional to the logarithm of the number of eliminated distinctions.

Moreover, constraint preservation prevents the transformation from relaxing into an equilibrium distribution. Active control must be applied to maintain the constraint manifold, and this control necessarily dissipates energy into the environment. Since both effects are unavoidable under the stated conditions, $\Delta S(T) > 0$. $\square$

This theorem has direct consequences for any computational model that relies on lossless reconciliation, perfect rollback, or reversible merge. Such models may be useful idealizations, but they cannot be realized without externalizing entropy costs.

Remark 2. *Logical reversibility does not imply physical reversibility. The former is a property of symbolic descriptions; the latter is constrained by thermodynamic law.*

The thermodynamic framing also clarifies why storage-centric systems eventually fail at scale. Persisting state is not free: it requires ongoing energy expenditure to stabilize physical representations and preserve interpretability over time. As systems grow in size and heterogeneity, the entropy cost of maintaining globally consistent stored state grows superlinearly, eventually dominating all other considerations.

From this perspective, abandoning storage as a primitive is not an aesthetic choice but a physical necessity. Computation must be re-conceived as an irreversible process that operates within entropy budgets rather than as a reversible manipulation of persistent objects.

5 From Files to Semantic Locality

If storage is no longer treated as a primitive, the question immediately arises: where, then, does meaning reside? The answer proposed here is that meaning does not inhere in artifacts at all, but in the space of admissible transformations that relate artifacts to one another under constraint.

A file, taken in isolation, carries no intrinsic meaning. Its interpretation depends on surrounding assumptions: type systems, schemas, conventions of use, implicit contracts between agents, and background knowledge that is rarely encoded explicitly. What appears as a stable object is in fact a temporary cross-section through a far richer semantic process.

This observation is familiar in practice. The same stored representation may be interpreted differently by different agents, or by the same agent at different times. Compatibility is therefore not a property of artifacts themselves, but of the contexts in which they are embedded. When a stored object appears to "break," it is not because its bits have changed, but because the constraints that once made it interpretable are no longer satisfied.

To account for this, we replace the artifact-centric view of meaning with a locality-centric one. A semantic locality is the region within which meanings can be stably negotiated, transformed, and repaired without external intervention. Outside such a locality, coherence cannot be assumed and must be actively reconstructed.

Semantic locality determines what questions can be asked, what transformations are intelligible, and what reconciliations are possible. It is therefore prior to any notion of state or snapshot. What appears as a "current state" is always a projection relative to a chosen locality.

Definition 2 (Semantic Locality, Informal)**.** *A semantic locality is a bounded region of interpretive stability within which transformations preserve meaning under shared constraints and acceptable entropy cost.*

This definition emphasizes three features. First, locality is bounded: no system admits global semantic transparency. Second, locality is constraint-relative: meaning is preserved only insofar as shared constraints remain satisfied. Third, locality is thermodynamically limited: transformations that exceed an entropy budget fall outside the locality, even if they are logically conceivable.

The transition from files to semantic localities therefore represents a shift from static representation to dynamic admissibility. Meaning is not something stored and retrieved, but something maintained through ongoing constraint satisfaction.

6 Formal Structure of Semantic Localities

The informal notion of semantic locality can be made precise by introducing a minimal formal structure that captures its essential features without committing to unnecessary implementation detail.

Definition 3 (Context Space)**.** *A context space is a tuple $\mathcal{C} = (S, \mathcal{T}, \vdash, \Delta)$, where S is a set of semantic states, $\mathcal{T}$ is a set of transformations $S \to S$, $\vdash$ is a satisfaction relation between states and constraints, and $\Delta : \mathcal{T} \times S \to \mathbb{R}_{\geq 0}$ assigns an entropy cost to each transformation when applied to a given state.*

Semantic states should be understood abstractly: they may represent configurations, interpretations, commitments, or summaries of event histories. The formalism does not require that states be fully explicit or globally accessible.

Definition 4 (Semantic Locality)**.** *A semantic locality is a context space equipped with a coherence predicate* $\mathrm{Coh} : S \to \{0, 1\}$ *such that the following conditions hold. First, coherence is preserved under admissible transformations: if* $\mathrm{Coh}(s) = 1$ *and $t \in \mathcal{T}$ is admissible at s, then* $\mathrm{Coh}(t(s)) = 1$*. Second, satisfied constraints are preserved: if $s \vdash C$, then $t(s) \vdash C$. Third, entropy production is bounded: there exists $\varepsilon > 0$ such that $\Delta(t, s) < \varepsilon$ for all admissible pairs (t, s).*

These conditions encode the intuition that a locality is not merely a region of logical consistency, but a region of sustainable semantic activity. Transformations that preserve constraints but exceed the entropy budget are excluded, reflecting the fact that semantic repair is limited by available resources.

One immediate consequence is that admissibility is not arbitrary.

Proposition 1. *For a fixed constraint set and entropy bound, the admissible transformation set $\mathcal{T}$ of a semantic locality is uniquely determined.*

Proof. Suppose $\mathcal{T}_1$ and $\mathcal{T}_2$ are two admissible transformation sets satisfying the locality conditions for the same $(S, \vdash, \Delta, \varepsilon)$. Assume for contradiction that there exists a transformation $t \in \mathcal{T}_1$ with $t \notin \mathcal{T}_2$. Then t must violate at least one locality condition relative to $\mathcal{T}_2$. If it violates constraint preservation or coherence preservation, it cannot belong to $\mathcal{T}_1$ either, contradicting the assumption. If it violates the entropy bound, it exceeds ε and thus fails admissibility in both cases. Hence $\mathcal{T}_1 \subseteq \mathcal{T}_2$. Symmetry yields equality. $\square$

This result formalizes an important conceptual shift. Infrastructure does not *select* what transformations are allowed; it instantiates constraints whose logical and thermodynamic consequences determine admissibility uniquely.

It is also important to note what this framework does not assume. It does not require global state, total ordering of events, or universal agreement on interpretation. Local coherence suffices. Indeed, insisting on global coherence would violate the entropy bounds that define locality in the first place.

Remark 3. *Semantic locality generalizes the notion of scope in programming languages, consistency domains in distributed systems, and interpretive frames in human communication. In each case, meaning is stabilized only locally and temporarily.*

The formalization of semantic locality thus provides the foundation on which the remainder of the theory is built. Merge, infrastructure, agency, and automation will all be analyzed as phenomena that arise from interactions between partially overlapping localities under irreversible constraints.

7 Infrastructure as Constraint Space

Infrastructure is commonly described in instrumental terms. It is treated either as a repository in which artifacts are stored, or as a transport layer through which information flows. In both cases, infrastructure is conceived as a passive substrate that supports computation without shaping its semantics. This view is no longer tenable.

In practice, infrastructure functions as a system of constraints that determine which transformations are admissible, which interpretations are coherent, and which histories can be sustained. Type systems, schemas, protocols, access controls, organizational norms, and legal frameworks all operate in this way. They do not merely store or transmit information; they actively delimit the space of meaningful action.

To characterize infrastructure as constraint space is to recognize that its primary function is semantic regulation. Constraints do not merely prohibit certain transformations; they make others possible by stabilizing expectations and suppressing ambiguity. A programming language without a type system permits many transformations, but few that are intelligible or safe. A social system without norms permits many actions, but few that are mutually interpretable.

Constraint spaces are inherently temporal. Constraints are not imposed once and for all; they evolve slowly relative to the transformations they regulate. This temporal asymmetry explains why infrastructure appears inert despite exerting continuous influence. By the time a constraint becomes visible as an object of deliberation, it has already shaped countless local interactions.

From the perspective developed here, infrastructure is best understood as a field of slow-moving semantic forces. It shapes trajectories of transformation without specifying outcomes in advance. Computation unfolds within this field, and the cost of deviating from its constraints is paid in entropy.

This reconceptualization has a crucial implication: infrastructure cannot be replaced wholesale without incurring massive semantic loss. Because constraints encode accumulated compromises between interpretability, efficiency, and power, they cannot be discarded without collapsing the localities they support.

Remark 4. *Infrastructure is conservative not because it resists change, but because it embodies the entropy cost of past reconciliations.*

8 Merge Is Not an Operation, but a Physical Event

Within storage-centric models, merge is typically treated as an operation on artifacts. Two versions of a file are combined according to deterministic rules, and conflicts are treated as exceptional cases to be resolved or eliminated. This framing presupposes that the underlying histories are commensurable and that any inconsistency can, in principle, be resolved algorithmically.

From the perspective of semantic locality, this presupposition fails. Merge is not an operation on states but an event in which two locally coherent histories are forced into contact under a shared constraint regime. The outcome of this contact is not determined solely by the prior states, but by the constraints, entropy budgets, and interpretive judgments available at the moment of reconciliation.

Formally, consider two semantic localities $\mathcal{L}_1$ and $\mathcal{L}_2$ that overlap partially but not completely. Each locality supports a set of admissible transformations and maintains coherence relative to its own constraint set. A merge event occurs when a transformation is attempted that requires the joint satisfaction of constraints from both localities.

When the combined constraint set is satisfiable within the available entropy budget, a new locality may emerge that subsumes both. When it is not, coherence can only be restored by discarding or weakening some commitments. In either case, information is lost: either through abstraction, reinterpretation, or outright elimination of incompatible history.

This observation can be sharpened into a general impossibility result.

[Impossibility of Perfect Merge] There exists no merge process M that is simultaneously lossless, reversible, fully automated, and constraint-preserving for all admissible semantic states.

Proof. Assume, for contradiction, that such a merge process exists. Losslessness implies that all semantic distinctions present in the input states are preserved in the output. Reversibility implies that the inputs can be recovered from the output without additional information. Automation implies that the merge can be performed without external semantic judgment. Constraint preservation implies that all constraints satisfied by the inputs remain satisfied by the output.

Now consider two admissible states s_1 and s_2 whose constraint sets C_1 and C_2 are individually satisfiable but jointly inconsistent. Losslessness requires that the commitments corresponding to both C_1 and C_2 be preserved. Constraint preservation then requires that the merged state satisfy $C_1 \cup C_2$, which is impossible. Any attempt to weaken or discard constraints violates losslessness. Any attempt to prioritize one constraint set over the other requires external semantic judgment, violating automation. Hence no such merge process exists. $\square$

The significance of this theorem lies not in its formal content but in its interpretive consequences. Merge is inherently a site of semantic choice. The choice may be deferred, obscured, or distributed, but it cannot be eliminated.

This explains why merge is experienced, in practice, as a moment of friction. Conflicts are not anomalies to be engineered away; they are manifestations of irreducible semantic incompatibility. The effort required to resolve them reflects the entropy cost of reconciling divergent histories.

Remark 5. *Merge events are the semantic analogues of dissipative interactions in physical systems: they reduce degrees of freedom while increasing entropy.*

Understanding merge as a physical event rather than an algebraic operation clarifies why no amount of tooling can make it entirely painless. Tools can support local repair, but they cannot eliminate the underlying irreversibility.

9 Sheaf-Theoretic Semantics and Obstruction

The tension between local semantic coherence and global inconsistency admits a natural mathematical expression in the language of sheaf theory. The value of this formalism is not that it introduces category-theoretic sophistication for its own sake, but that it provides a precise vocabulary for reasoning about partial agreement, contextual validity, and irreducible conflict without presupposing global consistency.

At an intuitive level, a sheaf encodes the idea that semantic states are defined relative to contexts, and that moving between contexts necessarily involves restriction, forgetting, or reinterpretation. A local section represents a semantic commitment that is internally coherent within a given context. The problem of merge then becomes the problem of whether such local commitments can be consistently glued together.

Let $(X, \leq)$ denote a partially ordered set of contexts, ordered by inclusion or refinement. Larger contexts subsume smaller ones by imposing additional constraints or by enlarging the domain of interpretation.

Definition 5 (Semantic Presheaf)**.** *A semantic presheaf $\mathcal{F}$ on $(X, \leq)$ assigns to each context $U \in X$ a set $\mathcal{F}(U)$ of admissible semantic states, and to each relation $U \leq V$ a restriction map*

$$\rho_{VU} : \mathcal{F}(V) \to \mathcal{F}(U),$$

such that ρ_{UU} is the identity and $\rho_{WU} = \rho_{VU} \circ \rho_{WV}$ whenever $U \leq V \leq W$.

Restriction maps formalize the act of projecting a semantic commitment into a smaller or less expressive context. Information may be lost under restriction, but coherence is preserved by construction.

The presheaf becomes a sheaf precisely when compatible local sections can be uniquely glued.

Definition 6 (Gluing Condition)**.** *A semantic presheaf $\mathcal{F}$ satisfies the gluing condition if, for any family of contexts $\{U_i\}$ covering a context V, and any family of local sections $s_i \in \mathcal{F}(U_i)$ that agree on all overlaps $U_i \cap U_j$, there exists a unique section $s \in \mathcal{F}(V)$ such that $\rho_{VU_i}(s) = s_i$ for all i.*

In semantic systems of any realistic complexity, this condition frequently fails. Local commitments that are individually coherent may nonetheless conflict when compared on overlaps. Such failures are not bugs; they are structural features of distributed meaning.

[Obstruction to Global Semantic Coherence] Let $\mathcal{F}$ be a semantic presheaf and $\{U_i\}$ a covering family. If there exist local sections $s_i \in \mathcal{F}(U_i)$ such that

$$\rho_{U_i, U_i \cap U_j}(s_i) \neq \rho_{U_j, U_i \cap U_j}(s_j)$$

for some i, j, then no global section over $\bigcup_i U_i$ exists.

Proof. Consider the Čech complex associated with the covering $\{U_i\}$. A global section exists if and only if the family (s_i) lies in the kernel of the Čech coboundary map δ^0, which measures disagreement on overlaps. The stated inequality implies $(s_i) \notin \ker(\delta^0)$, hence no global section exists. $\square$

The failure of gluing is thus a cohomological obstruction. Importantly, this obstruction is not eliminable by local refinement alone. It reflects a genuine incompatibility in semantic commitments, not a lack of expressive power.

From this perspective, merge is not the act of producing a global section where none exists, but the act of modifying, weakening, or discarding local sections so as to reduce the obstruction. This reduction necessarily incurs entropy cost, since it eliminates distinctions that were previously maintained.

Remark 6. *Global coherence is not a default condition. It is a rare and resource-intensive achievement.*

10 Event-Historical Computation

Once semantic states are understood as local sections rather than global objects, the inadequacy of snapshot-based computation becomes apparent. A snapshot purports to represent the system "as it is," but in doing so it suppresses the history of transformations and the contexts under which those transformations were admissible.

Event-historical computation rejects snapshots as primitives. Instead, it treats events—irreversible transformations under constraint—as the fundamental units of computation. What is commonly called a state is then understood as a derived summary of an event history, constructed relative to a particular interpretive context and for a particular purpose.

Formally, let $\mathcal{H}$ denote a space of admissible event histories, partially ordered by prefix. Each history encodes not merely a sequence of transformations, but the constraints under which those transformations occurred.

A state function is then a projection

$$\pi : \mathcal{H} \times \mathcal{C} \to S,$$

where $\mathcal{C}$ denotes a context of interpretation. Different contexts induce different projections from the same history, and no single projection preserves all information.

This formulation makes explicit that state is neither unique nor complete. It is a lossy compression optimized for local action rather than global truth. The same history may yield multiple incompatible states depending on which constraints are foregrounded.

Event-historical computation aligns naturally with the thermodynamic perspective developed earlier. Each event increases entropy, and projections necessarily discard information about microstructure and alternative possibilities. Attempts to reconstruct prior states from projections encounter the same irreversibility that governs physical processes.

This perspective also clarifies why logs, traces, and audit histories play such a central role in contemporary systems. They are not secondary artifacts, but the primary carriers of computational meaning. Snapshots exist only to facilitate temporary coordination.

Remark 7. *A system that treats snapshots as authoritative must continually fight entropy. A system that treats history as primary works with it.*

The event-historical view thus completes the displacement of storage as a primitive. Computation proceeds not by updating a persistent state, but by accumulating irreversible commitments whose coherence must be actively maintained within local semantic bounds.

11 Integration with Entropic Field Frameworks

The formalism developed thus far admits a natural interpretation in terms of entropic field dynamics. This interpretation is not an optional analogy, but a consequence of treating semantic structure, transformation, and entropy as co-evolving quantities rather than as separable layers.

Recall that a semantic locality is characterized by a set of admissible states, a family of constraint-preserving transformations, and an entropy budget. These elements correspond directly to field-theoretic quantities. Semantic states may be understood as configurations of a scalar field encoding semantic density or commitment. Admissible transformations correspond to constrained flows on this field. Entropy production corresponds to dissipation induced by curvature, friction, or constraint enforcement.

From this perspective, infrastructure defines the geometry of the semantic field. Constraints carve out admissible regions of configuration space; transformations trace trajectories within this space; merge events correspond to regions of high curvature where incompatible flows intersect. The impossibility of perfect merge is then a geometric fact: there is, in general, no smooth continuation that satisfies all constraints simultaneously.

This field-theoretic view clarifies why semantic repair is local rather than global. Just as physical fields relax by redistributing energy locally rather than by instantaneously reconfiguring the entire domain, semantic systems restore coherence through localized adjustments that dissipate entropy. Global reorganization is both energetically prohibitive and semantically destabilizing.

Moreover, the event-historical perspective aligns naturally with field dynamics. An event corresponds to a localized reconfiguration of the field, constrained by boundary conditions imposed by infrastructure. The accumulation of events traces a trajectory through semantic configuration space, whose irreversibility reflects the dissipation inherent in constraint maintenance.

Seen in this light, computation is not the execution of discrete instructions, but the continuous evolution of a constrained semantic field punctuated by irreversible events. Storage-centric abstractions obscure this continuity by reifying transient projections as stable objects.

Remark 8. *Infrastructure functions as semantic curvature: it does not determine outcomes, but it shapes the paths along which evolution can proceed.*

12 Agency as Constraint Navigation

Within the framework developed here, agency is not a primitive property but an emergent capacity. A system exhibits agency to the extent that it can navigate constraint space in a way that preserves local coherence over time while selectively projecting future trajectories.

This conception departs sharply from representational models of intelligence. An agent need not possess an explicit global model of its environment. Indeed, such a model would be unusable in systems characterized by semantic locality and irreversibility. What matters instead is the ability to maintain viable semantic localities while responding adaptively to perturbations.

Formally, consider an agent embedded in a semantic field with a given constraint structure. The agent's actions correspond to selecting admissible transformations that keep the system within a region of coherence while steering it toward preferred configurations. Preference here should not be understood teleologically or psychologically, but as a bias in the selection of trajectories that minimize future entropy production or constraint violation.

Agency thus consists in the capacity to perform selective projection under entropy. The agent does not enumerate all possible futures; it projects a sparse subset of futures that are locally sustainable. Futures that exceed the entropy budget or violate constraints are not considered, not because they are logically impossible, but because they are physically or semantically inaccessible.

This framing dissolves the distinction between computation and cognition. Cognitive processes are instances of semantic field navigation under constraint. Learning corresponds to the reshaping of admissible trajectories; decision-making corresponds to the selection of locally coherent continuations.

Proposition 2. *Any system capable of sustaining a semantic locality across nontrivial perturbations exhibits minimal agency.*

Proof. Sustaining a semantic locality requires selecting transformations that preserve coherence and respect entropy bounds despite perturbations. This selection cannot be reduced to passive dynamics; it requires discriminating between admissible and inadmissible trajectories. Such discrimination constitutes minimal agency. □

This notion of agency is compatible with both biological and artificial systems, and does not presuppose consciousness, representation, or intentionality in the philosophical sense. It is a structural property of systems that maintain themselves in constraint space over time.

13 Limits of Automation Revisited

The preceding analysis allows the limits of automation to be stated with greater precision. Automation succeeds when the relevant semantic activity can be contained within a stable locality whose constraints and entropy budget are well-characterized. Within such regions, admissible transformations can be enumerated, optimized, and executed reliably.

However, automation fails at the boundaries between localities. Merge events, interpretive shifts, and constraint reconfiguration necessarily involve irreducible ambiguity. These situations require choices that cannot be derived from the system's internal state alone, because the constraints themselves are under negotiation.

This is not a contingent limitation of current technology, but a structural one. Any attempt to fully automate semantic reconciliation would require encoding external judgment into the system, thereby reintroducing precisely the semantic dependence automation was meant to eliminate.

Human judgment enters here not as a mysterious faculty, but as a source of additional constraints. Humans supply context that is not locally available, thereby enabling new localities to be formed. In thermodynamic terms, they act as external reservoirs capable of absorbing or supplying entropy.

Remark 9. *Automation does not eliminate judgment; it merely displaces it to the boundaries of semantic locality.*

This reframing explains both the power and the fragility of automated systems. They perform extraordinarily well within stable regimes, and catastrophically fail when forced to operate beyond them. The appropriate response is not to pursue ever more global automation, but to design systems that recognize and respect the boundaries of their semantic competence.

14 Future Directions

The framework developed in this essay opens several lines of inquiry that merit systematic investigation. These directions are not peripheral extensions but natural consequences of abandoning storage as a primitive and reconceiving computation as irreversible semantic evolution.

One open problem concerns the computational complexity of reconciliation under bounded entropy. While it has been shown that perfect merge is impossible in general, the structure of approximate merge remains poorly understood. In particular, it is not yet clear under what conditions local reductions of cohomological obstruction yield globally adequate behavior, or how the cost of such reductions scales with the size and heterogeneity of the underlying constraint space.

A related direction involves the dynamics of semantic locality formation and collapse. Semantic localities are treated here as given, but in practice they are constructed, eroded, and reconfigured over time. Understanding how constraints stabilize local coherence, and how entropy accumulation leads to locality failure, would provide a more complete dynamical theory of infrastructure evolution.

The relationship between semantic locality and learning also warrants deeper analysis. Learning can be interpreted as the reshaping of admissible transformations under revised constraints. From this perspective, overfitting, concept drift, and catastrophic forgetting correspond to failures of locality maintenance rather than to deficiencies in representation. Formalizing this connection could unify learning theory with the thermodynamic and sheaf-theoretic structures developed here.

Finally, the extension of this framework to self-modifying systems raises fundamental questions about stability and control. Systems that alter their own constraints risk destabilizing the very localities that make coherent action possible. Identifying conditions under which self-modification preserves long-term semantic viability is likely to be essential for the design of autonomous systems operating at scale.

These directions suggest that computation after storage is not merely a new architectural pattern, but a broad research program whose implications extend across theoretical computer science, distributed systems, and the study of intelligence.

15 Conclusion

This essay has argued that storage-centric models of computation no longer provide an adequate foundation for understanding modern computational systems. Files, snapshots, and persistent state are artifacts of a reversible abstraction that conceals the physical, semantic, and social costs of computation as it is actually practiced.

By treating computation as an irreversible, constraint-preserving semantic process, a different picture emerges. Meaning is not stored and retrieved, but maintained against entropy. Infrastructure is not a neutral substrate, but a field of constraints shaped by past reconciliations. Merge is not an algebraic operation, but a physical event in which incompatible histories are forced into contact, and something must be lost.

The formal apparatus developed here—semantic localities, entropy bounds, sheaf-theoretic obstruction, and event-historical computation—provides a unified language for reasoning about these phenomena without appealing to global state or perfect consistency. Local coherence replaces global truth; admissibility replaces execution; maintenance replaces computation as run.

This shift has consequences beyond software architecture. It reframes intelligence as constraint navigation rather than representation, agency as selective projection under entropy rather than optimization over fixed state spaces, and automation as a local acceleration bounded by irreducible judgment.

Storage, in this view, is revealed as a historical scaffold: a useful simplification under specific material conditions, but not a fundamental feature of computation. As those conditions have changed, the scaffold has become a liability, obscuring the real structure of semantic evolution.

Computation after storage is therefore not a rejection of rigor, but a demand for greater honesty. It insists that computation be understood as it is realized: in time, under constraint, with irreversible consequences. Infrastructure is not merely tooling. It is physics.

A Semantic Localities and Constraint Dynamics

Definition 7 (Constraint System)**.** *A constraint system is a pair $(C, \models)$ where C is a set of constraints and $\models \subseteq S \times C$ is a satisfaction relation.*

Definition 8 (Context Space)**.** *A context space is a tuple*

$$\mathcal{C} = (S, \mathcal{T}, \vdash, \Delta),$$

where S is a set of semantic states, $\mathcal{T}$ is a set of partial transformations from S to itself, $\vdash \subseteq S \times C$ is a satisfaction relation between states and constraints, and $\Delta : \mathcal{T} \times S \to \mathbb{R}_{\geq 0}$ assigns a nonnegative entropy cost to the application of a transformation at a given state.

Definition 9 (Admissible Transformation)**.** *Given $\varepsilon > 0$, a transformation $t \in \mathcal{T}$ is admissible at state $s \in S$ if:*

$$\forall c \in C,\ s \vdash c \Rightarrow t(s) \vdash c \quad \textit{and} \quad \Delta(t, s) \leq \varepsilon.$$

Definition 10 (Semantic Locality)**.** *A semantic locality is a context space together with a coherence predicate*

$$\mathrm{Coh} : S \to \{0, 1\}$$

such that admissible transformations preserve coherence.

[Closure of Admissibility] If t_1 is admissible at s and t_2 is admissible at $t_1(s)$, then $t_2 \circ t_1$ is admissible at s.

Proof. Constraint preservation follows by transitivity of $\vdash$. Entropy boundedness follows from subadditivity of Δ. □

[Locality-Induced Semigroup] For any semantic locality, the set of admissible transformations forms a semigroup under composition.

Proof. Associativity follows from function composition. Closure follows from the previous lemma. □

Proposition 3 (Uniqueness of Admissibility)**.** *For fixed $(S, \vdash, \Delta, \varepsilon)$, the admissible transformation set is uniquely determined.*

Proof. Any transformation violating constraint preservation or entropy bounds is excluded. All remaining transformations are admissible by definition. □

B Entropy Bounds and Irreversibility

Definition 11 (Semantic Macrostate)**.** *A semantic macrostate $s \in S$ corresponds to an equivalence class of physical microstates with cardinality $\Omega(s)$.*

Definition 12 (Entropy of a Semantic State)**.** *The entropy associated with a semantic state s is*

$$S(s) := k_B \ln \Omega(s).$$

Definition 13 (Entropy Production of a Transformation)**.** *For a transformation $T : S \to S$, the entropy production at state s is*

$$\Delta S(T, s) := S(T(s)) - S(s) + S_{\text{diss}}(T, s),$$

where $S_{\text{diss}}(T, s) \geq 0$ denotes dissipated entropy.

[Many-to-One Entropy Increase] If T is many-to-one, then there exists $s \in S$ such that

$$\Omega(T(s)) \geq \Omega(s).$$

Proof. Since T identifies distinct semantic states, their corresponding microstate classes are merged under T. □

[Landauer Lower Bound] If T erases n bits of semantic distinction, then

$$\Delta S(T, s) \geq n k_B \ln 2.$$

Proof. By Landauer's principle, erasure of one bit requires dissipation of at least $k_B T \ln 2$ of heat. Summing over n erased bits yields the bound. □

[Irreversibility of Constraint-Preserving Computation] Let $T : S \to S$ be a many-to-one transformation that preserves all semantic constraints satisfied by a state s and is physically realizable with finite energy. Then the associated entropy production satisfies

$$\Delta S(T, s) > 0.$$

Proof. By the many-to-one property, T erases semantic distinctions. Constraint preservation prevents relaxation to equilibrium. Combined with the Landauer lower bound, strict entropy increase follows. □

[No Reversible Merge] Any merge transformation that is many-to-one and constraint-preserving is thermodynamically irreversible.

C Merge Impossibility and Undecidability

Definition 14 (Merge Operator). *A merge operator is a partial function*

$$M : S \times S \to S$$

defined on pairs of semantic states for which reconciliation is attempted.

Definition 15 (Constraint Preservation). *A merge operator M preserves constraints if, for all $s_1, s_2 \in S$ and all constraints $c \in C$,*

$$s_1 \vdash c \;\wedge\; s_2 \vdash c \;\Rightarrow\; M(s_1, s_2) \vdash c.$$

Definition 16 (Lossless Merge). *A merge operator M is lossless if there exists an injective map*

$$\iota : S \times S \hookrightarrow S$$

such that all semantic distinctions present in (s_1, s_2) are preserved in $M(s_1, s_2)$.

Definition 17 (Automated Merge). *A merge operator M is automated if it is computable without access to external semantic information beyond its inputs.*

Definition 18 (Reversible Merge). *A merge operator M is reversible if there exists a computable inverse*

$$M^{-1} : \mathrm{Im}(M) \to S \times S.$$

[Impossibility of Perfect Merge] No merge operator is simultaneously lossless, reversible, automated, and constraint-preserving.

Proof. Assume such an operator M exists. Consider states s_1, s_2 satisfying constraints C_1, C_2 respectively, where C_1 and C_2 are individually satisfiable but jointly inconsistent. Losslessness requires preservation of both constraint sets. Constraint preservation requires satisfaction of both constraint sets. This is impossible. Any resolution violates either losslessness, automation, or reversibility. Contradiction. □

Definition 19 (Optimal Merge Problem). *Given $s_1, s_2 \in S$ and constraint set C, determine whether there exists $M(s_1, s_2)$ minimizing semantic loss while satisfying all constraints in C.*

[Undecidability of Optimal Merge] The optimal merge problem is undecidable.

Proof. Reduction from the Post Correspondence Problem. Given an instance of PCP, encode partial solutions as semantic states and constraint satisfaction as PCP matching. An optimal merge exists if and only if the PCP instance has a solution. □

[Algorithmic Limits of Reconciliation] No general algorithm can compute optimal semantic reconciliation for all inputs.

D Sheaf-Theoretic Formalization

Definition 20 (Context Poset)**.** *Let $(X, \leq)$ be a partially ordered set whose elements represent semantic contexts, ordered by inclusion or refinement.*

Definition 21 (Semantic Presheaf)**.** *A semantic presheaf $\mathcal{F}$ on $(X, \leq)$ consists of:*

$$U \mapsto \mathcal{F}(U), \quad (U \leq V) \mapsto \rho_{VU} : \mathcal{F}(V) \to \mathcal{F}(U),$$

satisfying $\rho_{UU} = \mathrm{id}$ and $\rho_{WU} = \rho_{VU} \circ \rho_{WV}$ for all $U \leq V \leq W$.

Definition 22 (Compatible Family)**.** *A family $(s_i) \in \prod_i \mathcal{F}(U_i)$ is compatible if*

$$\rho_{U_i, U_i \cap U_j}(s_i) = \rho_{U_j, U_i \cap U_j}(s_j) \quad \forall i, j.$$

Definition 23 (Sheaf Condition)**.** *A presheaf $\mathcal{F}$ is a sheaf if for every covering $\{U_i\}$ of V and every compatible family (s_i), there exists a unique $s \in \mathcal{F}(V)$ such that $\rho_{VU_i}(s) = s_i$.*

Definition 24 (Čech Complex)**.** *Given a covering $\{U_i\}$, define*

$$C^0 := \prod_i \mathcal{F}(U_i), \quad C^1 := \prod_{i<j} \mathcal{F}(U_i \cap U_j).$$

Definition 25 (Coboundary Operator)**.** *The coboundary map $\delta^0 : C^0 \to C^1$ is defined by*

$$(\delta^0 s)_{ij} = \rho_{U_i, U_i \cap U_j}(s_i) - \rho_{U_j, U_i \cap U_j}(s_j).$$

[Obstruction to Global Section] A global section exists over $\bigcup_i U_i$ if and only if

$$(s_i) \in \ker(\delta^0).$$

Proof. Immediate from the definition of the sheaf condition. □

Definition 26 (Obstruction Class)**.** *The obstruction class of (s_i) is its cohomology class*

$$[(s_i)] \in H^1(X, \mathcal{F}).$$

[Merge as Obstruction Reduction] Merge corresponds to selecting sections minimizing the norm of the obstruction class in $H^1(X, \mathcal{F})$.

E Event-Historical Semantics

Definition 27 (Event)**.** *An event is a triple (s, t, s') where $s, s' \in S$ and $t \in \mathcal{T}$ such that $t(s) = s'$ and t is admissible at s.*

Definition 28 (History)**.** *A history is a finite sequence of events*

$$h = (e_1, e_2, \ldots, e_n)$$

such that the target state of e_i equals the source state of e_{i+1}.

Definition 29 (History Space)**.** *Let $\mathcal{H}$ denote the set of all admissible histories, partially ordered by the prefix relation $\preceq$.*

Definition 30 (Prefix Order)**.** *For histories $h, h' \in \mathcal{H}$, define*

$$h \preceq h' \iff h \textit{ is a prefix of } h'.$$

[Prefix Poset] $(\mathcal{H}, \preceq)$ is a partially ordered set.

Proof. Reflexivity, antisymmetry, and transitivity follow directly from properties of finite sequences. □

Definition 31 (State Projection)**.** *A state projection is a function*

$$\pi : \mathcal{H} \times \mathcal{C} \to S$$

mapping histories and contexts to semantic states.

[Non-Injectivity of Projection] For any nontrivial history space $\mathcal{H}$ and context $\mathcal{C}$, $\pi(\cdot, \mathcal{C})$ is not injective.

Proof. Distinct histories may differ only in events whose effects are discarded under projection. □

[State as Lossy Summary] No state projection π preserves full historical information.

Proof. Assume π preserves full history. Then π is injective on $\mathcal{H}$, contradicting the previous lemma. □

Definition 32 (History Extension)**.** *For $h \in \mathcal{H}$ and admissible event e, define the extension*

$$h \cdot e := (e_1, \ldots, e_n, e).$$

[Irreversibility of Extension] There exists no computable inverse for history extension.

Proof. Extension strictly increases prefix order. No finite procedure recovers deleted alternatives. □

F Agency and Constraint Navigation

Definition 33 (Trajectory)**.** *A trajectory is a map*

$$\gamma : [0, T] \to S$$

such that for all $t \in [0, T)$ there exists an admissible transformation $t_t \in \mathcal{T}$ with

$$\gamma(t + \delta) = t_t(\gamma(t))$$

for sufficiently small $\delta > 0$.

Definition 34 (Viable Trajectory)**.** *A trajectory γ is viable if*

$$\mathrm{Coh}(\gamma(t)) = 1 \quad \forall t \in [0, T].$$

Definition 35 (Perturbation)**.** *A perturbation is a transformation*

$$p : S \to S$$

not necessarily admissible with respect to the original locality.

Definition 36 (Recovery Map)**.** *A recovery map is an admissible transformation*

$$r : S \to S$$

such that $\mathrm{Coh}(r(p(s))) = 1$ *whenever recovery is possible.*

Definition 37 (Minimal Agency)**.** *A system exhibits minimal agency if, for any viable state s and admissible perturbation p, there exists a recovery map r such that*

$$\mathrm{Coh}(r(p(s))) = 1.$$

[Selective Trajectory Maintenance] Minimal agency implies the existence of a selection mechanism over admissible trajectories.

Proof. Recovery requires discriminating between admissible and inadmissible continuations. □

[Agency Criterion] A system sustains a semantic locality under bounded perturbations if and only if it exhibits minimal agency.

Proof. Necessity follows from the definition of viability. Sufficiency follows from the existence of recovery maps preserving coherence. □

Definition 38 (Entropy-Bounded Navigation)**.** *An agent navigates constraint space if it selects trajectories γ such that*

$$\int_0^T \Delta(t_t, \gamma(t))\, dt \le E$$

for some finite entropy budget E.

[Agency Under Entropy Constraint] Entropy-bounded navigation implies sustained local coherence over finite time.

G Complexity, Scalability, and Semantic Limits

Definition 39 (Semantic Decision Problem). *A semantic decision problem is a tuple $(S, C, \mathcal{T})$ together with a query*

$$Q : S \to \{0, 1\}$$

whose evaluation must preserve all constraints in C under admissible transformations $\mathcal{T}$.

Definition 40 (Semantic Consistency Problem). *Given a finite set of semantic states $\{s_i\} \subseteq S$ and constraint set C, determine whether there exists a state $s^* \in S$ such that*

$$s^* \vdash C \quad \textit{and} \quad s^* \preceq s_i \ \forall i,$$

where $\preceq$ denotes refinement or extension.

Definition 41 (Semantic Merge Decision Problem). *Given $s_1, s_2 \in S$ and constraint set C, decide whether there exists $s^* \in S$ such that*

$$s^* = M(s_1, s_2) \quad \textit{and} \quad s^* \vdash C.$$

[Semantic Consistency is NP-Hard] The semantic consistency problem is NP-hard.

Proof. Reduction from Boolean satisfiability. Encode Boolean variables as semantic commitments and clauses as constraints. A satisfying assignment exists if and only if a consistent semantic state exists. □

[Semantic Merge is Undecidable] The semantic merge decision problem is undecidable in general.

Proof. Reduction from the halting problem. Encode program states as semantic states and termination as a constraint. A valid merge exists if and only if the program halts. □

Definition 42 (Local Consistency Radius). *For a semantic locality $\mathcal{L}$, the local consistency radius r is the maximum depth such that all constraints are satisfiable within r interaction steps.*

[Local Sufficiency] For any semantic system with bounded local consistency radius r, maintaining consistency within radius r implies practical global coherence under bounded interaction.

Proof. All constraint violations must manifest within r steps. Beyond this radius, violations are causally disconnected under admissible transformations. □

Definition 43 (Entropy Cost Function)**.** *Define the entropy cost of a reconciliation as*

$$E(M) := \sum_{i=1}^{n} \Delta(t_i, s_i)$$

for transformations t_i applied during merge.

[Superlinear Entropy Growth] For semantic systems with interaction graph G, entropy cost grows at least superlinearly in the size of the minimal separator of G.

Proof. Each separator edge introduces an independent constraint reconciliation cost. The number of such costs scales with separator size. □

[Scalability Limit] No semantic system enforcing global consistency can scale linearly in system size.

Definition 44 (Semantic CAP Property)**.** *A semantic system is said to satisfy the Semantic CAP property if it simultaneously exhibits the following four conditions: global semantic consistency (C), meaning that all observers agree on semantic state; availability (A), meaning that local semantic transformations are always permitted to proceed; partition tolerance (P), meaning that the system continues to operate under locality partitions; and semantic constraint preservation (S), meaning that all admissible transformations preserve the governing semantic constraints.*

[Semantic CAP Impossibility] No distributed semantic system can simultaneously satisfy C, A, P, and S.

Proof. Partition tolerance forces independent evolution. Availability forces acceptance of local transformations. Consistency and semantic preservation require global reconciliation, which is undecidable or entropy-divergent. □

Definition 45 (Semantic Scalability Regime)**.** *A system operates in a scalable semantic regime if it relaxes at least one of global consistency or semantic preservation.*

[Necessity of Locality] Semantic locality is a necessary condition for scalable computation.

References

[1] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.

[2] C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17(6):525–532, 1973.

[3] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630, 1957.

[4] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley, New York, 2nd edition, 2006.

[5] S. Mac Lane. *Categories for the Working Mathematician.* Springer, New York, 2nd edition, 1998.

[6] R. Ghrist. *Elementary Applied Topology.* Createspace, 2014.

[7] G. E. Bredon. *Sheaf Theory.* Springer, New York, 2nd edition, 1997.

[8] S. Abramsky and A. Brandenburger. The sheaf-theoretic structure of non-locality and contextuality. *New Journal of Physics*, 13:113036, 2011.

[9] B. W. Lampson. Hints for computer system design. *ACM SIGOPS Operating Systems Review*, 17(5):33–48, 1983.

[10] E. A. Brewer. Towards robust distributed systems. In *Proceedings of the 19th ACM Symposium on Principles of Distributed Computing*, 2000.

[11] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

[12] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[13] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

[14] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

[15] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[16] J. A. Barandes. Indivisible stochastic processes. *Physical Review A*, 102:052209, 2020.

[17] K. Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11:127–138, 2010.

[18] H. A. Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6):467–482, 1962.

[19] T. Winograd and F. Flores. *Understanding Computers and Cognition.* Addison-Wesley, 1986.