# What Collapse Should Forget: A Retention Calculus for Spherepop

Flyxion

July 2026

*Against the Kitchen Sink* left one objection unresolved on purpose. History-primitive architectures — Git, event sourcing, write-ahead logs, Raft's replicated log — keep getting rediscovered at the systems level while general-purpose languages remain stubbornly state-centric, and the essay's proposed explanation, that mainstream languages simply inherited state-primacy from hardware history rather than conceptual necessity, was conceded there to be too easy an answer. The harder question it left standing was narrower and more tractable than it first appears: Collapse, the operator that projects an accumulated history onto an observable state, was defined as a reconstruction step whose actual reduction rule was left uninstantiated, gestured at with the phrase *coarse-grained history compression* and never specified. This essay specifies it. The claim is that retention is not a matter of taste or architecture but an ordinary optimization problem, and that state, audit logs, and full history retention are not three different philosophies of computation but three points on a single curve, distinguished only by what a system expects to be asked later and how much it costs to be wrong.

## 1 Continuations, Queries, and Sufficiency

Fix a history $H$, an unbounded, growing sequence of tagged events of the kind the earlier essay's typing rules already produce: atoms marked *selected* by Pop, *rejected* by Refuse, narrowed in admissibility by Bind. The reason to retain any of this at all is that something in the future will need to read it back. Call that something a continuation, and formalize the space of things a continuation might need as a class of queries $Q$, where each $q \in Q$ is a function $q : \mathcal{H} \to A_q$ mapping a history to some answer in a codomain $A_q$ specific to that query — *what is the current balance*, *was there a withdrawal over a given threshold in the last month*, *what did this variable contain three steps ago*. A retention policy is a map $\varphi : \mathcal{H} \to \hat{\mathcal{H}}$ compressing a history into some smaller representation $\hat{H} = \varphi(H)$, and $\varphi$ is *sufficient* for $Q$ if every query in $Q$ can still be answered correctly from the compressed representation: for every $q \in Q$ there exists $\hat{q}$ such that $\hat{q}(\varphi(H)) = q(H)$ for every admissible $H$. This is a direct transplant of the classical statistical notion of a sufficient statistic, in which a statistic $T(X)$ is sufficient for a parameter $\theta$ exactly when nothing about $\theta$ that the full sample $X$ could tell you is lost by observing only $T(X)$ [1]. Ordinary variable assignment in a state-primitive language is the special case $Q = \{\, q_{\text{now}} \,\}$, where $q_{\text{now}}$ asks only for the current value: the minimal sufficient $\varphi$ for that single query is the current value itself, and everything else in the history is, correctly, thrown away. Spherepop's original insistence on retaining everything is the opposite special case, $Q$ unbounded and unspecified in advance: when any future query might be asked, the only sufficient $\varphi$ is the identity map, and full retention stops being an ideological commitment and becomes the mathematically forced answer to an unbounded query class.

## 2 Minimal Sufficient Histories

Between those two extremes lies the interesting territory, and it is worth being precise about what makes a retention policy good rather than merely sufficient. Sufficiency alone is a weak requirement, since the identity map is trivially sufficient for any $Q$; what a system actually wants is a *minimal* sufficient $\varphi$, the smallest representation that still answers everything in $Q$, in exact analogy with a minimal sufficient statistic in classical estimation theory, characterized by the Fisher–Neyman factorization criterion and unique up to a bijection among statistics of the same information content [2]. Size here should be measured the way the rest of this research program already measures it: as an entropy over the space of possible compressed representations, $R(\varphi) = \log |\operatorname{range}(\varphi)|$, the same option-space entropy that governs Pop's strict entropy descent in the base calculus. A minimal sufficient $\varphi^*_Q$ for a fixed, known $Q$ is therefore the retention policy that minimizes $R(\varphi)$ subject to $\varphi$ remaining sufficient for every $q \in Q$ — and this quantity is well defined and computable in principle whenever $Q$ itself is fixed and known in advance. The balance-only query above has a minimal sufficient statistic of a single running sum; a query class that additionally asks for the maximum single transaction in the current calendar month has a minimal sufficient statistic of a running sum plus a running maximum reset monthly; a query class that asks for the exact sequence of transactions within a rolling window has a minimal sufficient statistic of exactly that window, no more and no less. Each of these is a legitimate instance of Collapse, differently parametrized, and none of them is either ordinary mutable state or unbounded history retention — they are the calculus's actual middle, occupied by nearly every real system that keeps some history but not all of it.

## 3 When the Query Class Is Unknown: Retention as Rate–Distortion

The trouble, of course, is that $Q$ is essentially never fixed and known in advance. A system rarely gets a complete specification of every question it will ever be asked; at best it has a rough sense of which queries are likely, and how costly it would be to fail one. This is precisely the situation rate–distortion theory was built to formalize, and Collapse's retention rule is naturally stated as a rate–distortion problem once that translation is made [3, 4]. Let $\pi$ be a prior distribution over an anticipated query population, and let $d(q(H), \hat{q}(\varphi(H)))$ measure the cost of answering $q$ from the compressed history $\varphi(H)$ rather than the true history $H$ — zero when $\varphi$ still answers $q$ correctly, and some positive repair cost, possibly unbounded, when it cannot. Define the expected distortion of a retention policy as $D(\varphi) = \mathbb{E}_{q\sim\pi}\big[\, d\big(q(H), \hat{q}(\varphi(H))\big)\,\big]$ and its rate as the same entropy $R(\varphi)$ used above. The retention problem a real Collapse implementation actually faces is then the classical rate–distortion optimization: minimize $R(\varphi)$ subject to $D(\varphi) \le D_{\max}$, or equivalently, for a Lagrange multiplier $\lambda$ expressing the exchange rate between storage cost and expected repair cost,

$$\varphi^*_\lambda \;=\; \arg\min_{\varphi} \Big[\, R(\varphi) \;+\; \lambda\, D(\varphi) \,\Big].$$

The two extremes recovered in the previous section fall out of this single expression as limiting cases rather than as separately chosen philosophies. As $\lambda \to 0$, storage is cheap relative to the cost of a failed future query, and the optimum drifts toward retaining everything, recovering Spherepop's original unbounded-history default. As $\lambda \to \infty$, a failed future query is nearly free compared to the cost of storage, and the optimum drifts toward the smallest sufficient statistic for whatever query dominates $\pi$, typically the current value alone, recovering ordinary mutable state. Git, audit logs, and write-ahead logs sit at intermediate, deliberately chosen values of $\lambda$, and the reason

mainstream general-purpose languages default close to the $\lambda \to \infty$ end while these specific systems default much closer to $\lambda \to 0$ is no longer a mystery once $\pi$ is allowed to differ between them: a typical loop variable's query population is almost entirely $q_{\text{now}}$, so a language optimizing for the typical program is optimizing correctly, not accidentally, when it discards almost everything. A ledger's query population reliably includes rare but catastrophically expensive audit queries, so an accounting system optimizing for its own $\pi$ is equally correct to retain far more. Neither language design nor Git was wrong. They were solving the same optimization with different, and in both cases locally accurate, priors.

## 4 The Information Bottleneck Reading

There is a second, closely related formalization worth stating because it sharpens what *relevant* means in the sentence "retain what is relevant to future continuations." The information bottleneck method poses exactly this problem in a different but formally compatible vocabulary: given a source variable $X$ and a relevance variable $Y$, find a compressed representation $\hat{X}$ that minimizes the mutual information $I(X;\hat{X})$, a direct measure of representational cost, while maximizing the retained mutual information $I(\hat{X};Y)$ with the variable that actually matters [5]. Reading $X$ as the full history $H$ and $Y$ as the space of future query outcomes drawn from $\pi$, Collapse's retention policy is an information bottleneck compression of history with respect to the future, and the Lagrangian trade-off $R(\varphi) + \lambda D(\varphi)$ above is the same trade-off the bottleneck method optimizes, restated in the terms this calculus already uses elsewhere for entropy and admissibility. This reading matters less as a new result than as a demonstration that the retention problem, once stated formally, is not a bespoke difficulty Spherepop happens to face; it is an instance of a well-studied compression problem, which is itself a form of evidence that the underlying question — how much of the past must be kept to remain correct about the future — is a real and general one rather than an artifact of this particular calculus's vocabulary.

## 5 A Convergent Literature: State Abstraction

A third independent literature arrived at the identical structure from a completely different direction, which is worth noting for the same reason Git was worth dwelling on in the earlier essay: independent, uncoordinated convergence is stronger evidence than any single derivation. Research on state abstraction in sequential decision-making asks precisely when two distinct states of a Markov decision process can be safely merged, or when a smaller, coarser representation of the state space can be substituted for the full one without changing which future decisions are optimal [6]. A $Q_\pi$-irrelevance abstraction, in that literature's terms, merges two states exactly when they are indistinguishable with respect to every future decision under a given policy — which is a sufficiency condition in exactly the sense defined above, with the query class $Q$ replaced by the class of all future value computations a decision-making agent might need. Reinforcement-learning researchers did not set out to formalize Collapse, and Spherepop was not built with Markov decision processes in mind, but both fields, under real pressure to compress an unwieldy representation without breaking whatever the representation was for, rediscovered the same criterion: keep exactly what distinguishes futures that matter, and nothing that does not.

## 6 A Worked Example

Concreteness disciplines a formalism that risks staying too abstract to evaluate. Consider a server logging a stream of financial transactions, each an amount and a timestamp, and suppose three queries populate $\pi$ with different weights. The first, $q_{\text{now}}$, asks only for the current balance, and is asked constantly; its minimal sufficient statistic is a single running sum, updated on each transaction and otherwise costless to maintain. The second, $q_{30}$, asks whether any transaction exceeded a regulatory threshold within the trailing thirty days, asked rarely but carrying a severe repair cost when it cannot be answered, since failing to produce a required record during an audit is not a bug but a compliance failure with a real financial penalty; its minimal sufficient statistic is the running balance together with the exact transaction record for a rolling thirty-day window, since nothing coarser than the individual transactions within that window can answer a threshold query about them. The third, $q_{\text{deep}}$, asks for the exact transaction sequence from an arbitrary point arbitrarily far in the past, asked almost never and, when it is asked, usually satisfiable through a secondary channel such as a cold backup rather than the live system; its distortion cost is high in principle but discounted heavily by $\pi(q_{\text{deep}})$, which is small, and by the availability of a cheaper repair path than perfect retention. A rate–distortion-optimal $\varphi^*_\lambda$ for this $\pi$ is neither the balance alone nor the full unbounded history: it retains the running sum unconditionally, retains full transaction detail for exactly the trailing thirty days demanded by $q_{30}$'s much higher $\lambda$-weighted repair cost, and collapses everything older into a coarser monthly summary, relying on the cheaper backup channel to absorb the residual risk from $q_{\text{deep}}$'s small but nonzero probability. This is not a description of an idealized system; it is a description of how most real transaction-logging infrastructure is already built, arrived at independently of this formalism by engineers responding to the same cost structure rather than deriving it from a calculus. The formalism's contribution is not the retention window itself, which practitioners already know to build, but the demonstration that the window is the provably optimal answer to a specific, statable optimization problem rather than an engineering convention with no principled floor or ceiling.

## 7 Repair Costs and the Limits of Anticipation

None of this removes risk; it relocates and prices it. The entire construction depends on $\pi$, the anticipated distribution over future queries, and $\pi$ is a forecast, not a fact. When a real future query falls outside the support of $\pi$ — a compliance regime changes and begins asking for detail the thirty-day window never anticipated, a debugging session needs the exact state five months before anyone thought to log it at that resolution — the retention policy that was optimal against the assumed $\pi$ becomes, after the fact, the wrong answer, and the distortion the model priced in advance is realized as an actual, uninsured repair cost: reconstructing what was never kept, or admitting that it cannot be reconstructed at all. This is not a flaw specific to Spherepop's formalization; it is the general and unavoidable exposure of any compression scheme to a shift between the distribution it was tuned against and the distribution it is actually asked to serve, and rate–distortion theory has never claimed to make that exposure disappear, only to make the trade-off explicit and the residual risk a chosen, priced quantity rather than an invisible one. What the formalism changes is not whether a system can be caught short by an unanticipated query — it always could be, under any architecture — but whether the system's designer can say, in advance and precisely, what class of futures the current retention policy is and is not prepared for, which is a strictly stronger position than the silent, undocumented assumption every state-primitive language currently makes about its own $\pi$ without ever stating it.

## 8 What This Resolves, and What It Does Not

The fifth objection in the earlier essay asked why history-native architectures keep appearing at the systems level while languages themselves remain state-centric, and offered, provisionally, the idea that history-primacy might be a genuinely higher-level concern that a general-purpose language cannot cheaply provide as a universal default. This essay sharpens that provisional answer into something closer to a demonstration. Mainstream languages default near the state-only end of the $\varphi^*_\lambda$ curve not because they inherited an arbitrary accident of hardware history, and not because history-primacy was rejected on its merits, but because the query population a typical variable actually faces is overwhelmingly dominated by $q_{\text{now}}$, and a retention policy tuned to that population is optimal for it, exactly as this calculus would recommend. Git, audit logs, and event-sourced architectures occupy a different, deliberately chosen point on the same curve because their designers, correctly, assumed a $\pi$ in which provenance queries carry weight that a typical program variable's $\pi$ does not. Both defaults are locally rational answers to the same optimization problem under different, and in both cases largely accurate, priors, which means the earlier essay's instinct to treat history-primacy and state-primacy as two competing ontologies was already slightly the wrong frame; they are two settings of the same dial. What this essay does not resolve, and does not pretend to, is how a system should choose or revise $\pi$ itself, particularly under adversarial or simply unforeseeable shifts in what the future turns out to ask — that is a learning problem, an estimation problem under a moving target, and very plausibly the actual successor question this line of work has been circling since the retention question was first left open. It is a narrower, more tractable question than *should computation be history-based*, and for exactly that reason it seems like the right place to stop and start again.

## References


[1] Ronald A. Fisher. On the Mathematical Foundations of Theoretical Statistics. *Philosophical Transactions of the Royal Society A*, 222:309–368, 1922.

[2] Erich L. Lehmann and George Casella. *Theory of Point Estimation*, 2nd edition. Springer, 1998.

[3] Claude E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[4] Toby Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Prentice-Hall, 1971.

[5] Naftali Tishby, Fernando C. Pereira, and William Bialek. The Information Bottleneck Method. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing*, pages 368–377, 1999.

[6] Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a Unified Theory of State Abstraction for MDPs. In *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, pages 531–539, 2006.