Hierarchical Summarization with Oracle-Preserving Guarantees: Aligning Human Preferences with Data Extraction Pipelines

Mitchell Linegar

Abstract

Hierarchical summarization lets practitioners process documents that exceed an LLM's context window, but it also obscures whether the resulting summaries still encode the task value that analysts care about. We provide a probabilistic process-control framework for decomposable tasks (event extraction, counting, coding, and guided summaries with explicit rubrics) that estimates fidelity without rereading the full text. The core object is a trio of consistency conditions—Sufficiency, Merge Consistency, and Idempotence—that test only the inputs (leaves or summaries) accessible at each node. When these checks hold, the "oracle" value is preserved in expectation, and we can estimate the pipeline's empirical violation rate by sampling a fixed budget of leaves, internal merges, and leaf boundaries per audit run. We explicitly track how these rates must scale with the size of the hierarchy: driving the per-edge failure rate p low enough to keep $N \times p$ acceptable is the operational goal. The same signals also serve as optimization objectives: DSPy-style prompt programs can be bootstrapped by mining successful local merges and adding them as exemplars until the sampled violation rate falls below tolerance. Finally, when downstream supervision depends only on the oracle (e.g., DPO loss), training on audited summaries is information-equivalent to training on the raw documents. The result is a practical recipe for building, auditing, and tuning long-context pipelines that provide statistical guarantees about the process without requiring end-to-end verification of every document.

1 Motivation

Large corpora are indispensable in contemporary political science, but the objects we ultimately analyze are not the raw texts themselves. What researchers actually want are structured signals: whether an article reports a protest event, which actor sponsored an amendment, the date and location of a strike, or the ideological position of a party manifesto on decentralization or environmental policy. Text is rarely the endpoint. It is the raw material from which we hope to extract something analytically useful, and the central methodological question is how to move from messy, unstructured documents to the clean, interpretable variables that enter our regressions and substantive analyses.

This challenge has driven decades of innovation, from hand-coding schemes to dictionary methods to topic models. More recently, large language models have emerged as a powerful tool for structured extraction, and their use is now ubiquitous across the social sciences. A striking example appears in recent work by Benoit et al. [2025], who use ensembles of LLMs to estimate party positions on key policy dimensions from manifestos written in dozens of languages. Their method asks LLMs to first summarize what a text says about particular issues, then to score the author's positions on those issues. The results correlate highly with expert surveys and, when applied to coalition policy declarations, align more closely with standard models of government formation than traditional hand-coded estimates. This is "natural language understanding" in the applied sense: using machines to replicate the outputs of human interpretive judgments, at scale, cheaply, and in practically any language.

Why the excitement? Qualitative methods that rely on human experts are costly and do not scale. Quantitative text-as-data approaches scale well but often rest on strong assumptions about what word frequencies or topic proportions actually capture. LLM-based methods promise something in between: the depth of traditional qualitative analysis combined with the scalability of computational approaches. But this promise comes with a methodological caveat. Even when LLM predictions are highly accurate in a classification sense, using them directly as input variables in downstream analyses can introduce bias and invalidate confidence intervals in ways that are not immediately apparent.

Recent work on Design-based Supervised Learning [Egami et al., 2024] makes this concrete. Even when prediction errors are small (say, 5–10% misclassification), naively substituting LLM-generated labels for ground-truth annotations can substantially bias downstream regression coefficients and produce confidence intervals with far less than nominal coverage. The problem is not that the LLM is "wrong" in some average sense, but that its errors may be systematic in ways that correlate with the very outcomes we are trying to explain. The DSL framework addresses this by combining surrogate labels with a smaller number of gold-standard human annotations, using a doubly-robust procedure that guarantees valid inference even when the surrogate is arbitrarily biased. If we want to use LLM outputs as data, we need formal methods for understanding and correcting for their relationship to the ground truth.

What both of these literatures point toward, albeit from different angles, is the idea of an underlying *oracle* judgment. In Benoit et al. [2025], the oracle is implicit: the expert survey scores that serve as the benchmark against which LLM estimates are validated. In Egami et al. [2024], it is explicit: the gold-standard labels that anchor the debiasing procedure. In both cases, the fundamental object of interest is not the LLM output per se, but some latent "true" value that the researcher actually cares about and that expensive human judgment can, in principle, recover.

We formalize this intuition by positing an oracle f^* that maps a full document x to the task value $f^*(x) \in \mathcal{Y}$ that the researcher actually cares about. In this view, LLMs are useful only insofar as they preserve (or at least accurately predict) the information contained in $f^*(x)$. Summaries, scores, and extracted variables are valid research objects only if they leave the oracle untouched. We therefore treat the summarizer g as a targeted compression operator: its sole job is to discard tokens that are provably irrelevant to f^* while preserving every bit of evidence needed for the oracle to reach the same verdict.[1] The oracle might be human annotation, a high-quality-but-expensive model, or a hypothetical infinite-context extractor.[2] It is often too costly or infeasible to apply f^* to every raw document, which is precisely why we rely on compressed representations $g(x)$ in the first place.

This framing provides a natural bridge to two broader literatures. First, we are asking for the summary to be a *sufficient statistic* for $f^*(x)$ in the classical sense from statistical decision theory [Blackwell, 1953]: once you have the summary, the original document provides no additional information about the task value. Any downstream analysis that depends only on f^* will give the same answer whether we use the summary or the full document. Second, we connect to work on *mergeable summaries* in database theory [Agarwal et al., 2013b], which asks when compact sketches of data can be combined without losing information about some target statistic. In the mergeable-summaries literature, one constructs explicit algebraic data structures (sketches) that preserve commutative statistics like counts or heavy hitters under merge operations. The key operation is a hand-designed merge function that combines two sketches while maintaining the invariant. Our setting differs in two ways: we work with string concatenation, which is non-commutative (order matters), and we seek to preserve arbitrary semantic oracles like event extraction or narrative flow rather than simple numeric aggregates. We therefore replace the explicit algebra with *learned* local consistency conditions that force a

[1]Put differently, $g(x)$ must be a sufficient statistic for $f^*(x)$. It may be terse or awkward, but it cannot omit the features that f^* inspects.

[2]In practice we approximate f^* with whatever expensive-but-accessible supervision we can afford: expert coders, deliberation-grade models, or hybrid workflows. The audit is therefore relative to that chosen oracle.

stochastic summarizer to behave *as if* it were a mergeable sketch. When the oracle happens to be symmetric (e.g., counting events where order is irrelevant), our conditions collapse exactly to standard mergeability; Appendix I makes this correspondence precise.

Two practical constraints motivate the hierarchical approach. First, many documents are longer than a model's context window. In practice, one partitions x into contiguous blocks, summarizes each block with an LLM, and recursively merges those summaries up a tree. Second, even when context is not the binding constraint, researchers often *choose* to summarize for cost, latency, or workflow reasons: storing compact "event summaries" that can be relabeled as coding rules evolve, or generating intermediate representations that multiple downstream tasks can query. Either way, hierarchical summarization changes the object of analysis, and a natural question arises: *How can we ensure this pipeline leaves $f^*(x)$ unchanged on average, without running the expensive oracle on the full text?*

We answer this question by introducing three consistency conditions (leaf-level sufficiency, on-range idempotence, and merge consistency) that can be checked locally, at each node of the summary tree, without ever processing the full document. When these conditions hold, the final summary preserves the oracle in expectation: the expected value of the oracle applied to the summary equals the oracle applied to the original document. When the conditions are violated, we can estimate how often and by how much. Crucially, these same conditions double as optimization objectives. Using DSPy [Khattab et al., 2024], we treat the summarization policy as a programmable system whose prompt parameters are updated until violations disappear. Because each check inspects only the immediate child summaries, we obtain cheap supervision signals without requiring new external labels. The audits that certify a pipeline also describe how to tune it.

The remainder of the paper develops this framework. Section 2 provides a high-level overview, introducing the three consistency conditions and explaining how local checks yield global guarantees in expectation. Section 3 makes this precise, describing the algebraic structure we are trying to approximate and stating the main theoretical results. Section 4 develops the probabilistic audit: a sampling-based procedure for estimating violation rates and certifying that a pipeline meets a given fidelity threshold. We conclude with empirical applications that demonstrate the framework in practice.

Throughout, we focus on decomposable, information-centric tasks: event extraction, coding tuples, structured aggregation, and guided abstractive summaries with explicit rubrics where the oracle space $\mathcal{Y}$ is well-defined on bounded spans of text. Open-ended creative summarization, where the oracle rewards prose quality directly, falls outside the scope of this paper. So too do settings where the LLM itself introduces systematic bias unrelated to the summarization process. If a model were politically slanted or subject to content filtering that distorts the underlying signal, even a perfect summarization pipeline would faithfully preserve a corrupted oracle, and additional diagnostics would be needed.

2 Framework Overview

2.1 Preservation Objective

Consider a researcher who hands us a sprawling report, novel, or code base and wants a short artifact that still supports the questions they care about. They already know how to grade answers when the full text is available—they can run a painstaking rubric, consult a subject-matter expert, or query a deliberation-grade LLM. This grading procedure is the *oracle* we want the hierarchy to preserve. The only reason summarization enters the story is that calling the oracle on every long document is too expensive, so we compress the document into smaller spans that can be checked locally while still behaving like the original text when fed back to the oracle.

Section 1 argued informally that hierarchical summarization is the only scalable route to this goal. We now slow down and make the objects explicit. The researcher specifies the oracle $f^* : \mathcal{X} \to \mathcal{Y}$ and asks the summarizer to act as a faithful, local proxy for f^* on every span the hierarchy touches. Because f^* can be prohibitively expensive to evaluate, we typically instantiate it with a high-quality but limited resource f (expert annotations, a trusted LLM, or a human-in-the-loop workflow).[3]

Definition 1 (Oracle)**.** *Let $\mathcal{X}$ denote the set of all finite token sequences with concatenation $\oplus$ and identity $\varnothing$. An* oracle *is a function $f^* : \mathcal{X} \to \mathcal{Y}$ mapping any document to the task value $f^*(x)$ that the researcher ultimately cares about.*

A summarizer $g : \mathcal{X} \rightsquigarrow \mathcal{X}$ is the compression primitive we audit. Every call to g is *local*: it sees at most a single leaf block or the concatenation of two child summaries. Composing these local calls yields the global hierarchical pipeline that shortens a book into a paragraph, yet the pipeline is entirely determined by how g behaves on small neighborhoods. If the local behavior aligns with what the oracle expects, the entire pipeline can safely swap summaries for raw text. This locality is also what makes the verification procedure introduced below tractable.

Definition 2 (Summarizer)**.** *A* summarizer *is a (possibly stochastic) map $g : \mathcal{X} \rightsquigarrow \mathcal{X}$. When g uses randomness (temperature, seeds), statements about $g(x)$ are interpreted in expectation over this randomness.*

Two desiderata drive the theory. First, the oracle should return the same answer on the raw text and on the stored summary, i.e., $f^*(x) = f^*(g(x))$ for every realized span. This is the "left panel" intuition from Figure 2: the oracle should not be able to tell whether it was handed the book or the book's summary. Second, extra "clean-up" passes should be inert, meaning $f^*(g(s)) = f^*(s)$ whenever s is itself a summary. Figure 1 captures this idempotence guarantee on the right. When both desiderata hold, we can freely substitute summaries for text and freely re-edit summaries without corrupting f^*.

2.2 Hierarchical Pipeline Construction

Operationally we partition the document into contiguous chunks $(b_1, \dots, b_k)$ that fit inside the model's context window and assign them to the leaves of a full binary tree T (Figure 2). Each leaf is summarized once to produce $s_i = g(b_i)$, and every internal node u receives the strings from its children, concatenates them, and calls the same summarizer again: $s_u := g(s_{u_L} \oplus s_{u_R})$. Traversing the tree bottom-up shrinks an entire book into a single paragraph-width string at the root.

During execution we only store the summaries s_u. The raw substrings $S(u)$ underneath a node exist conceptually so that we can compare the pipeline back to the document during a verification pass. Because g always sees at most two child summaries, every operation fits inside a bounded context window, making the "local" g literally the function invoked at each node and enabling the sampling-based checks formalized by the probabilistic audit (Definition 3).

Table 1 serves as a concise reference for the objects that appear throughout the paper. It names the realized blocks (b_i), the latent spans $S(u)$, the stored summaries s_u, and the multi-round outputs $Z^{(R)}(x)$. With this dictionary in place, the informal description matches Figure 2: the left panel shows the oracle-equivalent replacements we hope for, while the right panel of Figure 1 encodes the idempotence requirement that prevents drift during clean-up passes.

[3]Whenever we write f^* in the guarantees below, read it as "the oracle we would like to preserve." In deployments we approximate f^* with an accessible f that we trust enough to act as the auditing oracle. Any gap between f and f^* simply becomes measurement error in the audit.

Symbol	Meaning
$\mathcal{X}$	Set of finite token sequences (free monoid with concatenation $\oplus$ and identity $\varnothing$)
$f^*:\mathcal{X}\to\mathcal{Y}$	Oracle mapping strings to task values; $d_{\mathcal{Y}}:\mathcal{Y}\times\mathcal{Y}\to[0,\infty)$ scores discrepancies
$g:\mathcal{X}\rightsquigarrow\mathcal{X}$	Local summarizer (possibly stochastic) applied at every node of the hierarchy
x_0	Base document; realized leaves b_i are contiguous substrings of x_0
u	Node of merge tree T with ordered children u_L, u_R and latent span $S(u)$
s_u	Stored summary at node u ($s_u = g(b)$ at leaves; $s_u = g(s_{u_L}\oplus s_{u_R})$ internally)
r	Root node with latent span $S(r) = x$ and stored summary $s_r = Z^{(1)}(x)$
$Z^{(R)}(x)$	Rth-round summary ($Z^{(1)}(x) = s_r$, $Z^{(R+1)}(x) = g(Z^{(R)}(x))$)
$\mathbb{E}$	Expectation over g's randomness (all equalities are deterministic when g is deterministic)

Table 1: Notation recap for hierarchical summarization. The table extends Appendix A so the main text can stand alone.

2.3 From Pipeline to Local Laws

With the mechanics in place, we can ask what it would take for the hierarchy to behave exactly like the full document. The informal pipeline suggests a natural "global wish list": summarizing, concatenating, and re-summarizing should behave like an algebraic sketch whose query answers coincide with the oracle. Classical mergeable summaries in databases [Agarwal et al., 2013b] realize this wish when the statistics of interest are commutative (counts, heavy hitters). Text is far messier. Concatenation is non-commutative ($u \oplus v \neq v \oplus u$), the summarizer is stochastic, and the oracle may care about discourse that spans blocks. We therefore cannot assume that a global law such as $f^*(u\oplus v) = f^*(g(u)\oplus g(v))$ holds everywhere. Instead, we ask for a set of *local* laws on the realized leaves and merges. Figure 3 foreshadows these requirements by spotlighting the only three prompts we ever need. Our main theorem shows that these local checks are sufficient: if every call to g behaves, the entire hierarchy behaves, regardless of depth, tree shape, or optional clean-up passes.

2.4 Local Consistency Requirements

We now state the three local conditions that, when satisfied, guarantee the final summary preserves the oracle in expectation. Figure 3 is the running picture to keep in mind: the gray tree provides context, and the three colored spotlights show the precise prompts we can pose to the oracle.

Sufficiency. Panel (A) of Figure 3 zooms in on a single realized leaf: compare a raw block b_i with its first-pass summary s_i. Every leaf summary must therefore already be a trustworthy stand-in for its source span. This is the local analogue of asking whether the snippet still answers the research question. If a paragraph mentions three protest events, the summary must preserve all three; if a paragraph contains no relevant information, both the paragraph and its summary should map to the same neutral element in $\mathcal{Y}$. Formally, $g(b) \sim b$ for every realized leaf b, where $\sim$ denotes equivalence under the oracle metric.

Idempotence. Panel (B) captures the "clean-up" operation practitioners often add after the first pass: re-summarize a stored span s and confirm that the oracle stays put. These edits must be harmless with respect to the oracle: once information has been packed into a summary, rewriting that summary should not change the answer to the research question. Formally, $g(s) \sim s$ for any string s already in the range of g.

Merge consistency. Panel (C) contrasts the two admissible routes through a realized merge: the tree path that summarizes children before concatenation and the dashed joint path that summarizes the raw span directly. Once individual spans are trustworthy, we need to ensure that merging them does not introduce drift. Summarizing before concatenating must therefore be information-equivalent to concatenating before summarizing. That is, for any two spans u and v:

$$u \oplus v \;\sim\; g(u \oplus v) \;\sim\; g(g(u) \oplus g(v)).$$

The first equivalence says that jointly summarizing a span preserves its oracle value. The second says that pre-summarizing the parts before merging yields the same result as processing them together. When both hold, the order of summarization and concatenation becomes irrelevant to the oracle.

2.5 Verification Strategy and Guarantees

Once the failure modes are clear, the natural question is whether checking local behavior is enough to trust the whole tree. The answer is affirmative: because we prove that local consistency implies global preservation (Theorem 1 below), we can verify a hierarchy by inspecting a small number of realized nodes rather than re-running the oracle on the full document. We refer to this sampling-based verification routine as the *probabilistic audit.* Each probe fits inside the context window, yet the resulting failure-rate estimates expose how often the pipeline corrupts task values, decoupling the cost of quality control from document length.

Definition 3 (Probabilistic audit)**.** *Given a realized hierarchy, a probabilistic audit samples leaves, internal merges, and stored summaries; evaluates Conditions* (C1)–(C2) *on those spans using the oracle f that approximates f^*; and reports the empirical violation rates $\hat{p}_{suff}, \hat{p}_{assoc}, \hat{p}_{idem}$.*

Auditing via sampling. Figure 3 overlays the realized tree with the three spotlights we sample. Panel (A) draws realized leaves to estimate the sufficiency violation rate $\hat{p}_{\text{suff}}$. Panel (B) re-summarizes an on-range span such as s_{34} to estimate the idempotence failure rate $\hat{p}_{\text{idem}}$. Panel (C) samples internal edges and checks whether the two admissible merge routes agree, producing $\hat{p}_{\text{assoc}}$. Because every probe fits inside a standard context window, the audit scales by drawing more samples rather than by re-running the entire hierarchy. Section 4 formalizes the estimators and ties them directly to the guarantees below.

2.6 Applications: Optimization and Learning

These conditions do more than certify existing pipelines. Because the invariants are local, they can serve as objective functions for prompt engineering or fine-tuning. Rather than optimizing a model to “summarize well”—a vague objective—practitioners can optimize the model to satisfy merge consistency: ensure that $g(\text{summary}_A \oplus \text{summary}_B)$ yields the same extracted events as $g(\text{text}_A \oplus \text{text}_B)$. Frameworks like DSPy [Khattab et al., 2024] let us treat the summarization policy as a programmable system whose prompt parameters are updated until violations disappear. Because each constraint operates on a single edge of the tree, gradients or prompt-adjustment heuristics can be estimated via the same sampling procedure used for audits. The measurements that certify a pipeline also describe how to tune it.

Finally, this local view yields a clean connection to downstream learning. When supervision depends on documents only through their oracle values (a common factorization that covers proper scoring rules, coding tasks, and preference models), training on oracle-preserving summaries is information-equivalent to training on full documents. We develop this connection in Section 3.10, showing that if the hierarchy passes our audit, training a policy on the summaries achieves the same population optimum as training on the originals.

3 The Mechanism: From Algebra to Conditions

To make the guarantees of the previous section precise, we must define what it means for a summary to be "correct" in a formal sense. We begin with the mathematical setup, describe the algebraic ideal we strive for, and then state the consistency conditions and their consequences as theorems.

3.1 Setup: Strings, Oracles, and Metrics

We treat every document as a finite sequence of tokens. Let $\mathcal{X}$ denote the set of such strings, write $u \oplus v$ for concatenation, and let $\varnothing$ denote the empty string. The task of interest is represented by a fixed oracle $f^* : \mathcal{X} \to \mathcal{Y}$ that maps strings to task values (labels, structured tuples, embeddings, etc.). In many applications $\mathcal{Y}$ is best viewed as a *state of accumulated evidence*: a running list of actors, claims, or scores that grows as we read more text. We therefore assume (and later audit) that f^* is well defined on substrings and that $\mathcal{Y}$ carries a neutral element $\emptyset$ representing "no evidence yet," so that leaves with no signal map to $\emptyset$ while informative leaves update the evidence state.

The quality of a summary is measured in the oracle space via a (pseudo)metric $d_{\mathcal{Y}} : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$.

A strong locality assumption. Treating $\mathcal{Y}$ as an accumulated-evidence state is powerful but brittle: it requires every leaf summary to carry the sufficient statistics needed to evaluate any continuation. For narrative tasks the oracle on a substring can be undefined without extra latent variables (e.g., "Did the bill become law?" before the court decision is read). We therefore flag this "oracle on substrings" premise as an empirical hypothesis rather than an axiom. Section 4 spells out the concrete audit we run to stress-test it: we repeatedly probe the merge-consistency condition along realized boundaries and treat persistent violations as evidence that the task cannot be decomposed at the chosen block size. In practice we expect to pair these audits with domain-specific diagnostics and, when necessary, expand the summary format until it truly functions as a sufficient statistic for f^*.

Scope: decomposable tasks. Throughout we restrict attention to *decomposable* objectives where the oracle can be evaluated on bounded spans: coding tuples, structured extraction, counting, and guided abstractive summaries whose rubric specifies the state that must propagate. Open-ended "summarize this document however you like" goals fall outside our model because no finite $\mathcal{Y}$ suffices. Whenever practitioners apply the framework to guided summaries, they must pin down the rubric (actors, claims, stance, etc.) so that f^* is literally the function being preserved.

A summarizer $g : \mathcal{X} \rightsquigarrow \mathcal{X}$ is a (possibly stochastic) map. When g is stochastic, we evaluate its fidelity in expectation. To keep notation minimal, we write $\mathbb{E}[\cdot]$ to average over the internal randomness of g (seeds, temperature). A summary z is considered a valid substitute for document x if they are indistinguishable under the oracle metric:

$$d_{\mathcal{Y}}\big(f^*(z), f^*(x)\big) \approx 0.$$

3.2 Supervision and Factorization

Downstream learning signals depend only on the oracle. Let $S^\star : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ denote an arbitrary supervision signal (a score, reward, or loss) and let $\mathcal{A}$ be the action or label space. We assume this supervision factors through the task value.

Assumption 1 (Conditional factorization)**.** *There exists a measurable $\bar{Q} : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}$ such that for every action $a \in \mathcal{A}$,*

$$\mathbb{E}[S^\star(X, a) \mid f^*(X)] = \bar{Q}(f^*(X), a) \quad \textit{almost surely.}$$

This assumption covers standard classification (the loss depends only on the latent label), preference learning (the reward depends on the task semantics), and noisy annotators (the conditioning is on X). Once $f^*(X)$ is known, there is no additional supervision signal left in the raw phrasing.

3.3 Oracle Equivalence

The next concept lets us reason modulo oracle equivalence. It mirrors the standard "almost sure equality" in probability theory but is tailored to the task metric.

Definition 4 (Oracle equivalence)**.** *Two strings $u, v \in \mathcal{X}$ are* equivalent at the oracle*, denoted $u \sim v$, if their oracle outputs are indistinguishable:*

$$u \sim v \iff d_{\mathcal{Y}}\big(f^*(u), f^*(v)\big) = 0.$$

All subsequent conditions are stated modulo this relation. Because $d_{\mathcal{Y}}$ is a (pseudo)metric, $\sim$ is reflexive, symmetric, and transitive.

Propagating $\sim$ through concatenation requires a context-compatibility property of the oracle:

Assumption 2 (Context-compatible oracle)**.** *For every context $w \in \mathcal{X}$, $u \sim v$ implies both $w \oplus u \sim w \oplus v$ and $u \oplus w \sim v \oplus w$.*

Appendix G records structural conditions on g and f^* that guarantee Assumption 2. We state it here to make the dependency explicit: without context compatibility the equivalence relation would not be a congruence and local checks would fail to propagate. In deployments we treat Assumption 2 as falsifiable. The merge-consistency checks in Section 4 deliberately splice summaries into fresh contexts to detect when oracle values drift under concatenation; persistent failures trigger either larger leaf blocks or richer summary schemas.

3.4 Sufficiency and Stability

Definition 5 (Oracle sufficiency in expectation)**.** *The summarizer g is* f^*-sufficient in expectation *if $\mathbb{E}[d_{\mathcal{Y}}(f^*(g(x)), f^*(x))] = 0$ for all $x \in \mathcal{X}$.*

Definition 6 (Summary idempotence)**.** *The summarizer g is* summary-idempotent *if every string already in the range of g keeps its oracle under another call to g: for any random Z supported on* range(g)*,*

$$\mathbb{E}\big[d_{\mathcal{Y}}(f^*(g(Z)), f^*(Z)) \mid Z\big] = 0.$$

Definition 7 (Exact classes $\mathcal{G}^\star$ and $\mathcal{G}^\star_{\text{stab}}$)**.** *Let $\mathcal{G}^\star := \{g : d_{\mathcal{Y}}(f^*(g(X)), f^*(X)) = 0$ a.s.$\}$ and $\mathcal{G}^\star_{\text{stab}} := \{g \in \mathcal{G}^\star : g$ also satisfies Definition 6$\}$.*

Doob–Dynkin factorization implies $g \in \mathcal{G}^\star$ if and only if the oracle can be recovered as a measurable function of $g(X)$. In classical statistics this is exactly the definition of a sufficient statistic: the summary $g(X)$ captures all information about $f^*(X)$, so conditioning on $g(X)$ leaves no task signal behind. Stability simply says re-applying g cannot change that oracle. Together they give a two-step "oracle-preserving normal form" for any single span: the oracle cannot tell whether it saw the raw text, its first summary, or a cleaned-up version. Figure 1 visualizes the equalities enforced by these requirements.

$$\begin{tikzcd}
\boxed{X} \arrow[r, dotted] \arrow[d] & \boxed{g(X)} \arrow[r, dotted] \arrow[d] & \boxed{g(g(X))} \arrow[d] \\
f^*(X) \arrow[r, leftrightarrow, dashed, swap, "d_{\mathcal{Y}}\text{-equal if } g \in \mathcal{G}^\star"] & f^*(g(X)) \arrow[r, leftrightarrow, dashed, swap, "d_{\mathcal{Y}}\text{-equal by idempotence}"] & f^*(g(g(X)))
\end{tikzcd}$$

Figure 1: Oracle-preserving normal form for a single span. Condition (C1) forces $f^*(g(X))$ to equal $f^*(X)$; Condition (C2) keeps additional summaries inert so $f^*(g(g(X)))$ matches as well.

This normal form already delivers two of our three laws: leaf sufficiency (replace X by $g(X)$ without changing f^*) and idempotence (replace $g(X)$ by $g(g(X))$ without changing f^*). What it does *not* guarantee is that these substitutions commute with the *merge* operations that build a hierarchy. Even if every individual span satisfies Figure 1, the oracle can still drift when two spans interact unless concatenation respects the same equivalences. The next subsection therefore expands from this single-span picture to the local merge checks that keep the entire tree in an oracle-preserving normal form.

3.5 Document Decomposition

Long documents routinely exceed the model's context window, so we cut x into contiguous blocks $(b_1, \ldots, b_k)$ and summarize them along a realized full binary tree T. Every leaf stores the raw substring b_i plus its first-pass summary $s_i = g(b_i)$. For any node u we write $S(u)$ for the latent span of raw text underneath it; we only need this symbol when comparing back to the original document in global arguments. Operationally the model never touches $S(u)$ once we leave the leaves.

The only thing that moves up the tree is the stored summary s_u. Leaves set $s_u = g(b_i)$, and each internal node u reads the strings from its two children, concatenates them, and runs the summarizer again: $s_u := g(s_{u_L} \oplus s_{u_R})$. Traversing T bottom-up produces the one-pass output $Z^{(1)}(x) := g(\text{root}(T))$, and optional clean-up passes apply the same rule recursively via $Z^{(R+1)}(x) := g(Z^{(R)}(x))$.

Figure 2 locks in the notation. Lowercase b_i denotes the raw substrings carved out of the base document x_0; their summaries are $s_i = g(b_i)$. Every internal node carries two labels: $S(u)$ for the latent span and s_u for the actual string we store and propagate. The recursion $s_u = g(s_{u_L} \oplus s_{u_R})$ means parents only see (s_{u_L}, s_{u_R}), so every merge fits inside the context window.

Remark (well-formed blocks). Throughout we assume the partition $(b_1, \ldots, b_k)$ is fixed in advance, consists of contiguous substrings of the underlying document, and respects whatever discourse boundaries the downstream task requires. In practice this means that each block is small enough to fit in context but large enough to carry coherent snippets of evidence (e.g., full sentences or paragraph segments).

3.6 The Algebraic Ideal

If we were summarizing simple data structures, the requirements for g would be straightforward. Consider the case where f^* counts keywords. Concatenating spans simply adds their counts: $f^*(u \oplus v) = f^*(u) + f^*(v)$. To preserve this information, a summarizer g (acting as a "sketch") would need to support a merge operation that mimics addition. This is the essence of mergeable summaries in database theory [Agarwal et al., 2013a]:

$$f^*(A \cup B) \approx \text{Query}\big(\text{Merge}(S_A, S_B)\big),$$

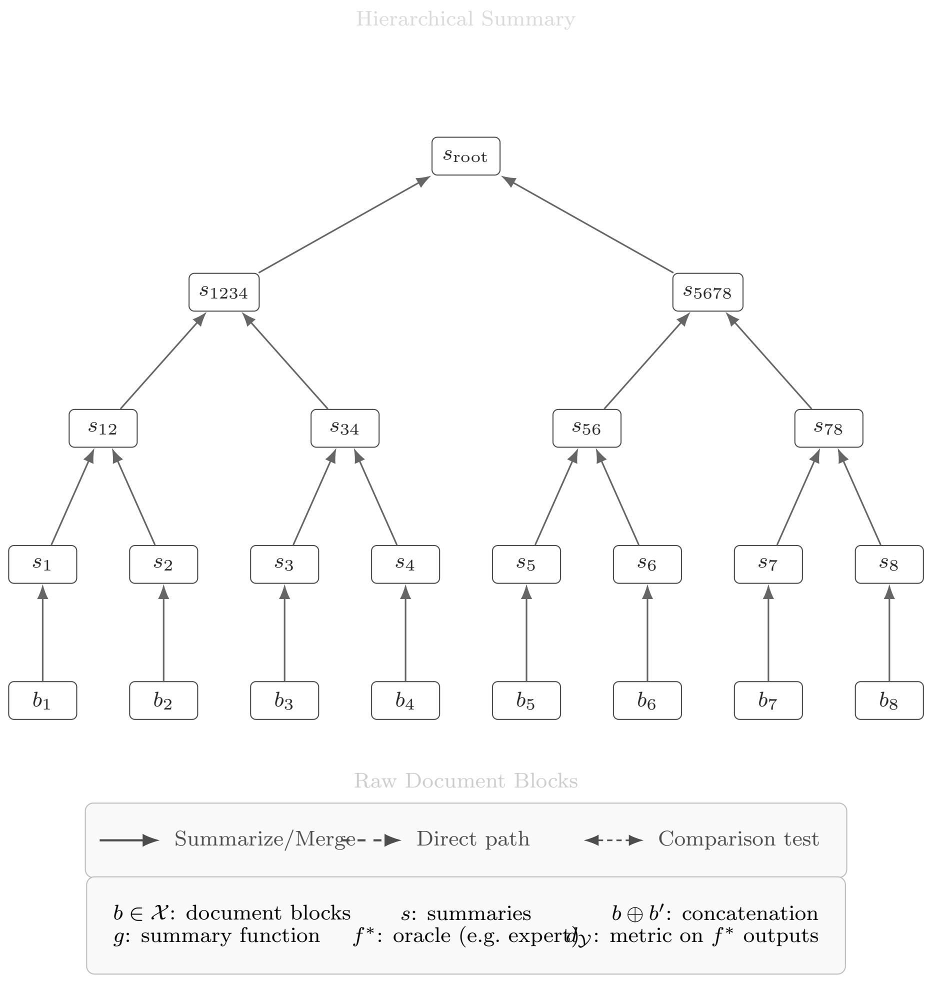
Hierarchical Summary
s_{root}
s_{1234}
s_{5678}
s_{12}
s_{34}
s_{56}
s_{78}
s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8
b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8
Raw Document Blocks
Summarize/Merge
Direct path
Comparison test
$b \in \mathcal{X}$: document blocks
s: summaries
$b \oplus b'$: concatenation
g: summary function
f^*: oracle (e.g. expert)
$d_{\mathcal{Y}}$: metric on f^* outputs

Figure 2: Hierarchical summarization as a binary tree. Each leaf block b_i is summarized once to produce $s_i = g(b_i)$. Internal nodes recursively summarize the concatenation of their children's stored strings, e.g., $s_{12} = g(s_1 \oplus s_2)$ and $s_{\text{root}} = g(s_{1234} \oplus s_{5678})$. The latent raw text under a node is denoted $S(u)$ for analysis, but the system only materializes the summaries s_u, making every merge auditable from bounded inputs.

where S_A denotes the stored sketch of set A and $\text{Merge}(S_A, S_B)$ produces a sketch of $A \cup B$.

For complex semantic tasks—event extraction, narrative summarization, or legal analysis—the merge operation is concatenation, which is non-commutative ($u \oplus v \neq v \oplus u$). Nevertheless, the algebraic intuition survives. Ideally, we would have

$$d_{\mathcal{Y}}\Big(f^*(u \oplus v),\ f^*\big(g(u) \oplus g(v)\big)\Big) = 0 \quad \text{for all } u, v \in \mathcal{X},$$

so that every hierarchy, schedule, or reduction behaves as a homomorphism with respect to f^*. If this global property held, hierarchical summarization would be trivially safe. However, large language models are not naturally associative algebraic operators. They are stochastic, sensitive to phrasing, and prone to drifting when context is segmented. We cannot assume this global structure exists; we must enforce it via local checks that approximate the desired algebra.

3.7 The Three Conditions: Formal Statement

Since we cannot verify the global property on massive corpora (which would require running f^* on the full text, defeating the purpose of summarization), we focus on properties that are local and auditable.

Convention 1 (Standing expectation convention)**.** *Expectations* $\mathbb{E}[\cdot]$ *at a node* u *are conditional on the realized subtrees below it. They average over the coin flips used to generate the summaries at* u *and its children.*

Condition 1: Sufficiency. The summary of every realized leaf must be a valid proxy for the raw block:

$$\forall \text{ realized leaf } b: \quad g(b) \sim b. \tag{C1}$$

Once this holds, a leaf summary may replace its source block anywhere because the two strings are equivalent at the oracle. If an LLM cannot satisfy this condition even on a single paragraph (e.g., by dropping a name), the hierarchy fails immediately. Panel (A) of Figure 3 isolates exactly this check: compare a realized block b_i with its summary s_i inside the oracle. Section 4 estimates the empirical violation rate of this condition by sampling leaves and checking how often the oracle disagrees.

Condition 2: Idempotence. Hierarchical pipelines often include optional clean-up passes that re-summarize a stored output to smooth the style or reduce length. For this to be safe, the summarizer must be a fixed point on its own range:

$$\forall \text{ summary } s \in \text{range}(g): \quad g(s) \sim s. \tag{C2}$$

Without this condition, repeatedly normalizing a string could cause task information to drift. When it holds, any extra style edit, deduplication sweep, or DSPy controller pass is provably inert at the oracle. Panel (B) of Figure 3 shows this "re-summarize and compare" test as a tangible audit. Appendix K.4 constructs an explicit hierarchy where Conditions (C1) and (C3) hold but g drifts the label under repeated summarization; Condition (C2) rules out exactly this failure mode.

Condition 3: Merge Consistency. The summarization pipeline must commute with concatenation modulo the oracle:

$$\forall u, v \in \mathcal{X}: \quad \underbrace{u \oplus v}_{\text{Raw}} \sim \underbrace{g(u \oplus v)}_{\text{Joint}} \sim \underbrace{g\big(g(u) \oplus g(v)\big)}_{\text{Disjoint}}. \tag{C3}$$

The first link asserts that jointly summarizing a span preserves its task value. The second link asserts that splitting the inputs before summarization is information-equivalent to processing them together. Panel (C) of Figure 3 shows how the two admissible routes align at a realized merge: the colored "tree path" summarizes children before concatenation while the dashed joint path summarizes the raw concatenation of b_{56}.

Remark 1 (Edge-wise compatibility)**.** *Audits often instantiate Condition* (C3) *at a single realized edge* (v, w) *by comparing the reference and pipeline routes:*

$$d_{\mathcal{Y}}\big(f^*(v \oplus w), f^*(g(g(v) \oplus g(w)))\big) = 0. \tag{E1}$$

Equation (E1) *probes the right link of the merge-consistency chain using quantities that fit inside the context window.*

3.8 Failure Modes

Once the pipeline is laid out, we can describe the most common ways it breaks. Each failure corresponds to a violation of one of the local conditions above and provides intuition for the audits we run later:

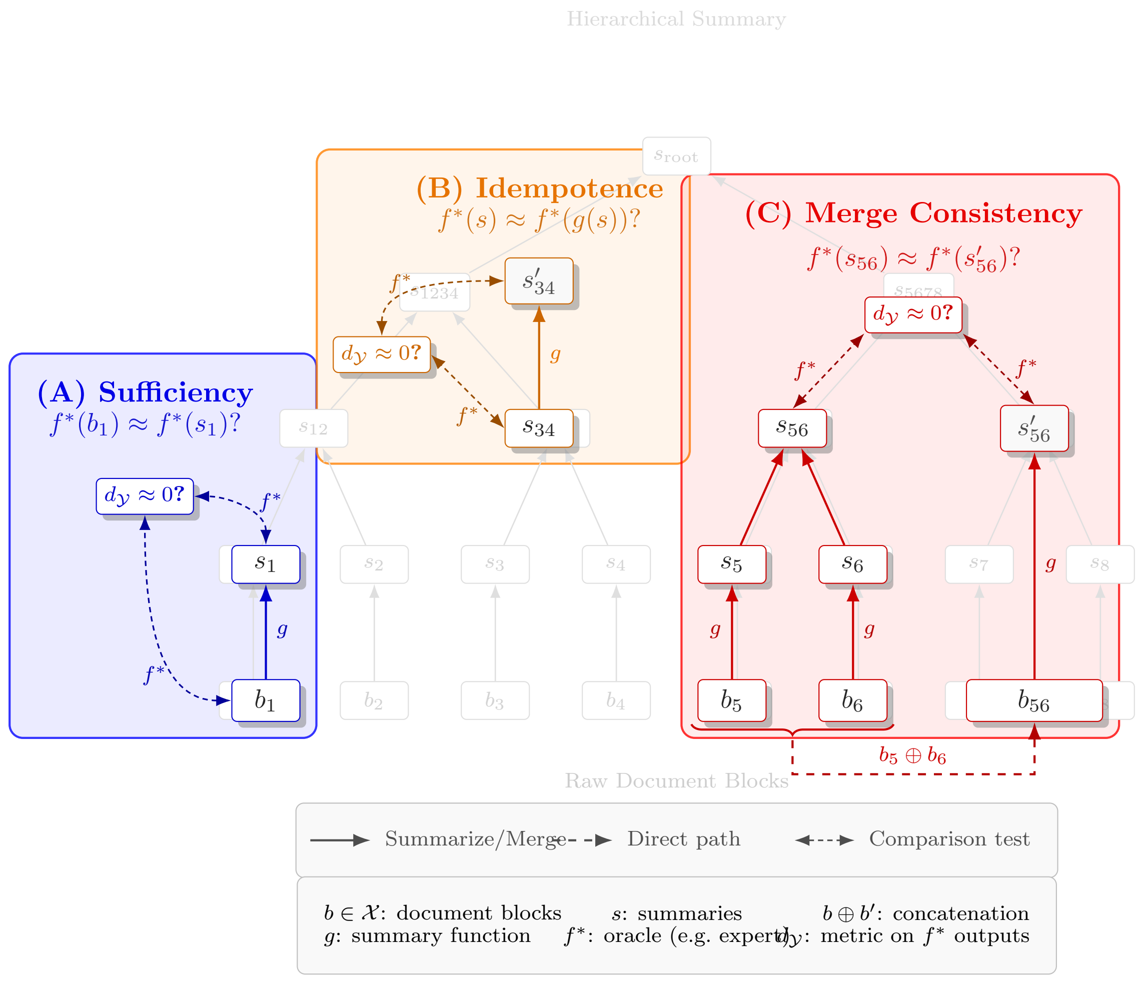
Hierarchical Summary
(B) Idempotence
$f^*(s) \approx f^*(g(s))$?
(C) Merge Consistency
$f^*(s_{56}) \approx f^*(s'_{56})$?
(A) Sufficiency
$f^*(b_1) \approx f^*(s_1)$?
s_{root}
s_{1234}
s_{5678}
s'_{34}
s_{34}
s_{12}
$dy \approx 0$?
f^*
g
s_{56}
s'_{56}
s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8
b_1 b_2 b_3 b_4 b_5 b_6 b_{56}
$b_5 \oplus b_6$
Raw Document Blocks
Summarize/Merge
Direct path
Comparison test
$b \in \mathcal{X}$: document blocks
s: summaries
$b \oplus b'$: concatenation
g: summary function
f^*: oracle (e.g. expert)
dy: metric on f^* outputs

Figure 3: Spotlight view of the three local conditions. (A) Leaf sufficiency compares a raw block b_i to its summary s_i. (B) Idempotence re-summarizes an internal span s_{34} and checks whether $f^*(s)$ drifts after calling g again. (C) Merge consistency contrasts the "tree path" summaries s_{56} with a direct summary of the joint raw span b_{56}. The gray tree anchors the realized hierarchy while the colored panels highlight the local prompts we can audit or optimize.

- **Leaf sufficiency failures.** A leaf summary omits or corrupts information the oracle needs. For example, a paragraph may mention three protest events but the summary records only two, so $g(b)$ no longer preserves $f^*(b)$.
- **Idempotence failures.** Optional clean-up passes that re-summarize stored outputs can drift the oracle value, turning precise tuples into vague prose if g does not treat its own outputs as fixed points.
- **Merge-consistency failures.** Even when leaves are accurate, the pipeline can hallucinate or drop details when it merges them. One route—summarize first, then concatenate—may deduplicate or reorder differently than the "summarize once at the merged span" route.

These modes appear whenever prompts are brittle or the summary schema is too terse. The rest of the section formalizes the checks needed to detect them and build pipelines that avoid them.

3.9 Consequences: Processing Invariance

The utility of these conditions is that they allow us to transport trust from the leaves to the root. We do not need to verify the final summary against the whole book; we only need to verify that each station of the assembly line is functioning correctly.

Theorem 1 (Inductive Preservation). *Fix a document partition $x = b_1 \oplus \ldots \oplus b_k$ and any full binary merge tree T. If C1 holds on every realized leaf and C3 holds at every realized internal node, then the final summary preserves the oracle in expectation:*

$$\mathbb{E}\left[d_{\mathcal{Y}}\big(f^*(Z^{(1)}), f^*(x)\big)\right] = 0.$$

Furthermore, if C2 holds, this preservation is maintained through any number of additional re-summarization rounds $Z^{(R)}$.

Proof: See Appendix B, Proof A.5.

Corollary 1 (Schedule invariance). *For any fixed partition $(b_1, \ldots, b_k)$, every full binary tree on these leaves yields the same expected oracle whenever Conditions* (C1) *and* (C3) *hold on its realized edges. Balanced reductions and daisy chains are therefore interchangeable.*

Proof: See Appendix B, Proof B.

Corollary 2 (Fold-of-folds invariance). *Consider any two-level plan that first reduces contiguous "folds" and then reduces the fold summaries. If Conditions* (C1) *and* (C3) *hold on every realized edge and Condition* (C2) *holds on the intermediate summaries, the composite schedule preserves f^* in expectation regardless of the within-fold or over-fold parenthesizations.*

Proof: See Appendix B, Proof B.

Together these results ensure that practitioners can pick whatever reduction schedule matches their infrastructure (balanced trees, daisy chains, fold-of-folds pipelines) without changing the expected oracle, provided the three conditions hold in the realized hierarchy.

Theorem 2 (Multi-round preservation). *Assume Conditions* (C1) *and* (C3) *hold along the realized tree so that Theorem 1 yields $\mathbb{E}[d_{\mathcal{Y}}(f^*(Z^{(1)}(x)), f^*(x))] = 0$. If, in addition, on-range stability* (C2) *holds, then $\mathbb{E}[d_{\mathcal{Y}}(f^*(Z^{(R)}(x)), f^*(x))] = 0$ for every $R \geq 1$ with $Z^{(R+1)}(x) := g(Z^{(R)}(x))$.*

Proof: See Appendix B, Proof B.

3.10 Safe Learning on Summaries

Assumption 1 ensures that downstream training signals depend on documents only through $f^*(X)$. When a hierarchy satisfies the three conditions, the oracle of each realized node matches the oracle of its source text, so any supervision computed on top of the hierarchy is statistically indistinguishable from supervision computed on full documents.

This information-equivalence only holds under Assumption 1; if rewards depend on surface style, summarization can distort the target regardless of the audit. Appendix C formalizes the reduction via Blackwell sufficiency, and Appendix F explains why we focus on Direct Preference Optimization (DPO) [Rafailov et al., 2023]: DPO is a practical way to align models with researcher preferences, its Bradley–Terry backbone matches our oracle-based factorization, and its loss is smooth enough to incorporate the condition checks as optimization objectives.

Once the audit certifies the tree (small $\hat{p}_{\text{suff}}$, $\hat{p}_{\text{assoc}}$, $\hat{p}_{\text{idem}}$), training on summaries achieves the same optimum as training on full text up to a gap controlled by the measured distortion. Because f^* may encode arbitrary (but expensive) researcher judgments, this route lets us align to those judgments directly; aesthetic or stylistic goals outside f^* remain out of scope.

3.11 Constructive Alignment via Local Bootstrapping

The consistency conditions are more than diagnostic tools; they describe an optimization problem. We view the summarization system as a programmable policy $\mathcal{P}_\theta$ whose parameters θ collect prompt instructions, few-shot exemplars, or DSPy controller settings. Because Conditions (C1) and (C3) evaluate only the strings already stored at child nodes, each audited edge yields an inexpensive supervision signal: a success certifies $(\text{input}, \text{summary})$ as a valid training pair, and a failure tells us exactly which constraint was violated.

In practice we maintain two lightweight modules. A `LeafSummarizer` maps raw blocks b to $s_b = g(b)$ and is tuned to satisfy Condition (C1). A `MergeSummarizer` maps stored child summaries (s_{u_L}, s_{u_R}) to s_u and is tuned to satisfy Condition (C3). A simple "consistency bootstrap" alternates between them:

i. Sample leaves and runs of the merge tree. For each leaf, run `LeafSummarizer`, evaluate $d_\mathcal{Y}(f^*(g(b)), f^*(b))$, and, when necessary, re-generate or request a human rewrite until the oracle agrees. Certified pairs (b, s_b) become part of the prompt context or fine-tuning buffer for the leaf module.

ii. Using the improved leaf module, generate child summaries (s_{u_L}, s_{u_R}), apply `MergeSummarizer`, check $d_\mathcal{Y}(f^*(g(s_{u_L} \oplus s_{u_R})), f^*(s_{u_L} \oplus s_{u_R}))$, and add successful triples to the merge module's training set. Because the check fits in context, we can perform this loop offline or online without touching the raw subtree.

iii. Continue sampling until the empirical failure rates estimated by the audit fall below the tolerance required for deployment. Additional human examples can be injected at any time; they are especially useful on structures where repeated regeneration never satisfies the oracle.

Appendix E provides a DSPy-oriented template for this loop. The main text only needs the structural fact: the same local comparisons that certify past runs also act as objective functions for the optimizer, allowing practitioners to tune θ with no extra labels beyond the oracle already assumed in our framework. Section 4 now turns these ingredients into the probabilistic audit that powers both certification and optimization in practice.

4 The Probabilistic Audit: Certifying Pipelines at Scale

The theoretical guarantees in Section 3 and the learning equivalences in Section 3.10 rely on the assumption that the sufficiency, merge-consistency, and idempotence conditions hold. In a real deployment, we cannot simply assume these properties; we must verify them. However, checking every node in a massive tree defeats the purpose of summarization.

We solve this dilemma using a *probabilistic audit.* Because Theorem 1 guarantees that global fidelity is structurally implied by local fidelity, we do not need to validate the final summary against the full text to monitor quality. Instead, we treat the summarization pipeline as a stochastic process and estimate the *Empirical Violation Rate* of each condition. A fixed sampling budget per audit run suffices to estimate these *rates*; certifying that a specific book is error-free still requires those rates to be extremely small.

Throughout we assume the atomic objects we audit—leaf blocks b and stored child pairs (s_{u_L}, s_{u_R})—are small enough that both the oracle f^* and the summarizer g can process them directly. The hierarchy is useful precisely because the root document X may be orders of magnitude larger than this audit budget; we audit the bounded pieces and rely on the structural results to propagate guarantees upward.

The spotlight overlay from Figure 3 is therefore both the conceptual map and the literal set of prompts we reuse during auditing: we repeatedly sample windows that look like Panels (A)–(C) and record whether the oracle agrees.

Interpreting the spotlights. Figure 3 makes the sampling logic concrete. Panel (A) draws a random realized leaf (here b_3) and invokes f^* on both b_3 and $g(b_3)$ to decide whether that block preserves the oracle, directly contributing to $\hat{p}_{\text{suff}}$. Panel (B) samples stored summaries (e.g., s_{34}), runs g again, and checks $f^*(g(s_{34}))$ against $f^*(s_{34})$ to estimate $\hat{p}_{\text{idem}}$. Panel (C) samples an actual boundary (b_5, b_6), runs both admissible links of Condition (C3) that fit in context, and compares the oracle of the raw joint span to the oracle of the disjoint summary path; every agreement reduces $\hat{p}_{\text{assoc}}$. Each spotlight therefore identifies a bounded context in which we can literally run f^* on both arguments, record a Bernoulli outcome, and keep the rest of the tree ghosted out because guarantees propagate from these audited edges to the unseen portions.

4.1 The Audit Protocol

The audit reports three sample means: the sufficiency violation rate ($\hat{p}_{\text{suff}}$), the merge-consistency violation rate ($\hat{p}_{\text{assoc}}$), and the idempotence violation rate ($\hat{p}_{\text{idem}}$).

Sampling Leaves (Sufficiency Condition). We sample n_ℓ leaf blocks $\{b_i\}$ uniformly from the corpus. For each block, we generate its summary $g(b_i)$ and compare the oracle values:

$$\hat{p}_{\text{suff}} := \frac{1}{n_\ell} \sum_{i=1}^{n_\ell} \mathbb{I}\left\{d_{\mathcal{Y}}\big(f^*(g(b_i)), f^*(b_i)\big) > \tau\right\},$$

where τ is a tolerance threshold (usually 0). This statistic is the empirical violation rate of Condition (C1). The check is cheap because leaf blocks are by definition short enough to fit in the model context.

Sampling Merge Consistency (The Structural Check). Condition (C3) requires a chain of equivalences, but we rarely have enough context to verify the entire chain at every node. Instead, we audit whichever link fits in memory for the sampled pair, always conditioning on the realized structure.

- **Case A: Leaf boundaries (Substitution).** When u, v are adjacent raw blocks, $u \oplus v$ fits in context. We compare the joint and disjoint summaries (the second link in Condition (C3)):

$$I_{\text{bound}} := \mathbb{I}\Big\{d_{\mathcal{Y}}\big(f^*(g(u \oplus v)),\ f^*(g(g(u) \oplus g(v)))\big) > \tau\Big\}.$$

- **Case B: Internal merges (Compression).** When u, v are stored summaries, they are short. We therefore compare the raw concatenation of the stored strings to the one-shot summary (the first link in Condition (C3), with on-range inputs):

$$I_{\text{merge}} := \mathbb{I}\Big\{d_{\mathcal{Y}}\big(f^*(u \oplus v),\ f^*(g(u \oplus v))\big) > \tau\Big\}.$$

For a batch of sampled leaf boundaries $\mathcal{B}$ and internal merges $\mathcal{M}$, the merge-consistency violation rate is the weighted average

$$\hat{p}_{\text{assoc}} := \frac{\sum_{(u,v)\in\mathcal{B}} I_{\text{bound}}(u,v) + \sum_{(u,v)\in\mathcal{M}} I_{\text{merge}}(u,v)}{|\mathcal{B}| + |\mathcal{M}|}.$$

This estimator covers both parts of Condition (C3); the sampling procedure simply chooses whichever link fits in context, and together the two cases enforce the entire Raw $\sim$ Joint $\sim$ Disjoint chain.

Sampling Stability (Idempotence Condition). If the pipeline includes post-processing rounds, we sample n_s summaries z_k that have already passed through g and check if re-summarization alters the oracle:

$$\hat{p}_{\text{idem}} := \frac{1}{n_s} \sum_{k=1}^{n_s} \mathbb{I}\left\{ d_{\mathcal{Y}}\big(f^*(g(z_k)), f^*(z_k)\big) > \tau \right\}.$$

This is the empirical violation rate of Condition (C2).

These three sample means provide unbiased estimates of the violation probabilities. Because every check fits in context, teams can run them on demand, streaming new samples until the observed rate and its confidence interval fall below a deployment threshold or batching entire corpora for periodic reviews. The same indicators also drive the optimization loop in Section 3.11: when the audit detects a violation, we immediately know which local objective is being missed. Appendix D shows how to convert the measured rates into worst-case global bounds if needed; otherwise, practitioners simply keep sampling until the empirical error rate is acceptable.

Scaling with tree size. Let a realized hierarchy contain N leaves and $M = N - 1$ merges. A transparent union bound (Appendix D) then gives

$$\Pr\big[\text{root violation}\big] \;\leq\; N\, p_{\text{suff}} + M\, p_{\text{assoc}} + (R-1)\, p_{\text{idem}}, \tag{1}$$

where $p_{\text{suff}}, p_{\text{assoc}}, p_{\text{idem}}$ denote the true per-edge violation probabilities and R counts optional clean-up rounds. This $N \times p$ scaling is the price of safety: even tiny per-edge failure rates accumulate over thousands of nodes. The audit's job is therefore to drive each p low enough that the product $N \times p$ (and $M \times p$) stays within tolerance, or to change the decomposition until it does. When a target corpus makes this impossible, the framework surfaces that fact explicitly instead of hiding it in asymptotics.

It bears repeating that these statistics describe the *edge failure rates*, not the correctness of a specific document. If a document has N leaves and the sufficiency failure rate is p_{suff}, the probability that every leaf is correct is roughly $(1 - p_{\text{suff}})^N \approx e^{-Np_{\text{suff}}}$. For large N, the root summary is reliable only when the local error rates are correspondingly tiny. The audit therefore provides continuous process control rather than an end-to-end certificate for any single long document.

4.2 Connecting the Audit to Downstream Learning

In practice the most informative number is the empirical leaf violation rate. Each leaf check compares raw text to its first summary, so $\hat{p}_{\text{suff}}$ is a direct estimate of how often the pipeline corrupts ground truth before any merges occur. The merge and stability rates refine this estimate by revealing whether additional deviations are introduced higher in the tree. Teams therefore monitor the trio $(\hat{p}_{\text{suff}}, \hat{p}_{\text{assoc}}, \hat{p}_{\text{idem}})$, treating them as summaries of pipeline error: if the sufficiency rate is below tolerance and the subsequent stages rarely introduce new violations, the hierarchy can be trusted to preserve f^*.

Recall from Section 3.10 that we wish to train a DPO policy π on summaries Z instead of documents X. The fundamental risk is that the policy will learn to exploit artifacts of the summarization rather than the true user preferences. Appendix F shows that the resulting regret is Lipschitz in the oracle distortion, so any improvement in the empirically estimated rates—especially $\hat{p}_{\text{suff}}$—translates directly into a tighter alignment guarantee. The probabilistic audit therefore doubles as a diagnostic for downstream training: we simply keep sampling leaves and merges until the observed rate of violations falls below the acceptable tolerance, at which point the same estimates justify training on the summaries.

4.3 Optimizing the Pipeline

Section 3.11 already sketched how the audit doubles as supervision. Here we spell out the deployment loop. We instantiate two promptable modules (leaf and merge), treat their instructions as parameters θ, and update them using any programmatic prompting framework—DSPy [Khattab et al., 2024] is convenient because it exposes an optimizer over few-shot traces and temperature. This is a form of active learning or hard-negative mining: violations reveal exactly which examples to upweight next. The procedure is as follows:

1. **Collect leaf data.** Sample leaves, run the current leaf module, and evaluate Condition (C1). Successful generations (b, s_b) are appended to the module's context or fine-tuning buffer. When a failure occurs, we may re-generate at higher diversity, manually rewrite the span, or request a human edit. Either way, the comparison produces a labeled example at negligible cost because both arguments fit in context.

2. **Collect merge data.** Using the best available leaf summaries, sample adjacent pairs (s_{u_L}, s_{u_R}), run the merge module, and evaluate Condition (C3). Successful merges become demonstrations for future runs. If repeated attempts fail, we flag the edge for escalation to a human or a larger teacher model; those "hard negatives" are inserted as mandatory exemplars in the next optimization round.[4]

3. **Update and stop.** Re-optimize the prompt parameters (e.g., via DSPy's gradient-free controller) on the accumulated traces and resume sampling. Because the audit statistics are empirical, the loop can be run offline (optimize until the held-out failure rate drops below tolerance) or online (deploy while continuously sampling and updating). The stopping rule is "continue until the estimated violation rate is acceptable," at which point the same estimates justify deployment.

This "consistency bootstrap" requires no external supervision beyond the oracle already assumed by the audit. Appendix E expands each step into DSPy-ready pseudocode including trace management and temperature search.

Calibration vs. deployment. The audit statistics assume the summarizer g is fixed. In practice we therefore separate the *calibration phase*, during which prompts or weights are updated using the bootstrap loop, from the *deployment phase*, during which we freeze g and estimate $(\hat{p}_{\text{suff}}, \hat{p}_{\text{assoc}}, \hat{p}_{\text{idem}})$ on held-out data. Mixing adaptation and measurement on the same stream breaks the i.i.d. assumptions behind the confidence bounds unless additional martingale corrections are applied.

5 Conclusion and Outlook

We have presented a framework to solve the central dilemma of long-document processing: how to scale semantic extraction to massive corpora without losing the ability to verify the results. By treating hierarchical summarization not as a black-box compression but as an algebraic reduction, we allow practitioners to audit the fidelity of the entire pipeline by inspecting only its atomic parts.

Our approach rests on a semantic generalization of *mergeable summaries.* While the database literature relies on explicit, commutative sketches to preserve counts and frequencies, we substitute these with "learned" consistency conditions—Sufficiency C1, Idempotence C2, and Merge Consistency C3—that force stochastic LLMs to behave *as if* they were rigorous algebraic operators. The LLM never becomes truly associative; it merely mimics the merge algebra closely

[4]In practice we keep a small budget of high-capacity calls (or human hours) for these escalations. They both unblock the pipeline and supply the most informative prompt traces.

enough that the oracle cannot distinguish the two routes. This connects the flexibility of modern language models with the structural guarantees of randomized streaming algorithms.

The primary contribution is operational. We replace the impossible requirement of end-to-end verification (reading the whole library to check the summary) with a scalable *probabilistic audit*. By sampling realized leaves and internal merges, estimating their failure rates, and monitoring especially the leaf violations that compare summaries to raw text, practitioners obtain a live picture of how often the hierarchy corrupts the oracle. These empirical error rates make visible the precise trade-offs between block size, merge schedule, and accuracy, allowing social scientists to certify the reliability of their data extraction without needing to manually code a statistically significant fraction of the full corpus. The guarantees are about process quality: if the measured per-edge failure rates are small, the pipeline is trustworthy; for any specific document, reliability still follows the familiar $(1-p)^N$ scaling.

These guarantees extend beyond measurement to learning. We showed that under standard factorization assumptions (where supervision depends only on the oracle f^*), training Direct Preference Optimization (DPO) policies on validated summaries is information-equivalent to training on full documents. The gap between the ideal policy and the learned policy is controlled by the empirically estimated violation rates, so continued sampling tightens the alignment guarantee. Because the oracle can encode arbitrarily nuanced researcher preferences (provided they are expensive but finite to evaluate), this route gives a practical way to align LLM-generated summaries with whatever semantics matter, while explicitly acknowledging that stylistic alignment outside the oracle's scope must be handled separately.

The limits of this approach are clear. The strongest equivalences hold only when supervision strictly factors through the task oracle; if human annotators reward presentational features lost during summarization, the alignment may drift. Likewise, substitution consistency (Condition (C3)) can fail if summaries drop context that later merges rely on. Those cases show up as merge-consistency violations in the audit and must be addressed by prompt tuning or stronger models. Finally, on-range stability is substantive: without it, additional "clean-up" rounds can silently flip labels, a failure mode our audit is designed to detect.

Methodologically, the blueprint is simple but demanding: Build hierarchies that pass the edge-wise audit; optimize models to satisfy merge consistency; and report the estimated violation rates alongside codebooks and regression tables. The result is a pipeline whose guarantees align with the research design, whose failures are interpretable, and whose outputs can be trusted to mean what the social scientist claims they mean.

References

Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Transactions on Database Systems*, 38(4):28:1–28:40, 2013a.

Pankaj K Agarwal, Graham Cormode, Zengfeng Huang, Jeff M Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Transactions on Database Systems*, 38(4):1–28, 2013b.

Kenneth Benoit, Kevin Munger, Arthur Spirling, et al. Scalable methods for interpreting political texts using large language models. *American Journal of Political Science*, 2025. Forthcoming.

David Blackwell. Equivalent comparisons of experiments. *Annals of Mathematical Statistics*, 24 (2):265–272, 1953.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

Ralph A. Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In *Journal of Algorithms*, volume 55, pages 58–75, 2005.

A. P. Dawid. The geometry of proper scoring rules. *Annals of the Institute of Statistical Mathematics*, 59:77–93, 2007.

Naoki Egami, Musashi Hinck, Brandon M. Stewart, and Hanying Wei. Using imperfect surrogates for downstream inference: Design-based supervised learning for social science applications of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437.

Omar Khattab, Keshav Gupta, Colin Burns, Arman Cohan Razi, Pieter Abbeel, and Christopher Re. Dspy: Compiling declarative language model calls into self-improving pipelines, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Margaret E. Roberts, Brandon M. Stewart, and Dustin Tingley. Structural topic models for open-ended survey responses. *American Journal of Political Science*, 58(4):1064–1082, 2014. doi: 10.1111/ajps.12103.

Greg Ver Steeg and Aram Galstyan. The information sieve, 2016. URL `https://arxiv.org/abs/1507.02284`.

Erik Torgersen. *Comparison of Statistical Experiments.* Cambridge University Press, 1991.

6 Connections to Related Work

Our focus on local merge compatibility draws on, and extends, several established theoretical traditions. We view our framework as a semantic generalization of randomized sketching, justified by decision-theoretic sufficiency.

Mergeable Summaries and Sketches. The closest formal analogue to our work is the theory of *mergeable summaries* for data streams [Agarwal et al., 2013a]. In that setting, a summary (or sketch) S supports a binary merge operator $\oplus$ such that $S(A \cup B) \approx S(A) \oplus S(B)$, with error guarantees that hold under arbitrary merge trees. Our framework can be understood as generalizing this logic to the semantic, non-commutative domain of natural language. Specifically, when the task oracle f^* is symmetric (e.g., word counts or bag-of-words statistics) and the metric is Euclidean, our *Merge Consistency* condition (C2) is mathematically isomorphic to the definition of a mergeable summary. In this specific case, checking Condition (C3) is equivalent to verifying that the LLM acts as a valid adder for a Count-Min sketch [Cormode and Muthukrishnan, 2005]. However, text data is rarely commutative—the order of paragraphs changes the meaning of a narrative. Our contribution is to impose the rigorous discipline of mergeable summaries on these non-commutative operations, replacing the explicit algebra of sketches with "learned" local consistency checks that force a stochastic summarizer to behave *as if* it were a mergeable data structure. Appendix I formalizes this bridge, mapping our notation $(\mathcal{X}, \oplus, f^*, g)$ onto the streaming literature and proving that Condition (C3) coincides with mergeability when the oracle is symmetric.

Decision Theory and Blackwell Sufficiency. A second foundation is decision theory. Under proper scoring rules, the optimal Bayesian action depends only on the posterior of the target; reporting the truth is optimal [Gneiting and Raftery, 2007, Dawid, 2007]. Our Assumption 1 instantiates this principle: if supervision factorizes through $f^*(X)$, then f^* is a *sufficient statistic* for the population decision problem. Replacing X with a summary that preserves f^* leaves the Bayes risk unchanged. This is a task-specific version of Blackwell sufficiency [Blackwell, 1953, Torgersen, 1991], where f^* acts as the sufficient reduction. While Blackwell's theorem is often invoked abstractly, our framework provides an operational audit to verify that a specific summarization pipeline actually retains this sufficiency in practice.

Information-theoretic parallels. Our framework is grounded in sufficiency and mergeable summaries, but it also admits an information-theoretic interpretation: the summary acts as the sufficient component while the discarded text serves as an ancillary "remainder." For readers interested in that lens, Appendix H sketches how the decomposition ideas behind the *Information Sieve* of Steeg and Galstyan [2016] can be adapted to oracle-preserving compression, though this viewpoint is entirely optional for the main results.

Preference Learning and RLHF. Modern alignment techniques, such as Direct Preference Optimization (DPO), model preferences using Bradley–Terry–Luce likelihoods driven by a reward function [Bradley and Terry, 1952, Rafailov et al., 2023]. A standard assumption in this literature is that human preferences depend on the input only through its task-relevant semantics (the oracle value), not incidental phrasing. Under this factorization, our preservation results imply that training DPO on oracle-preserving summaries attains the same population optimum as training on full documents. Absent such preservation, empirical performance gains can be illusory: a learned policy might entangle with artifacts of a particular merge schedule—a failure mode our audit is specifically designed to detect. When readability or fluency preferences fall outside f^*, they must be modeled by a richer oracle or handled by separate alignment channels.

Global Structure vs. Local Checks. Finally, we contrast our approach with corpus-level abstractions like Latent Dirichlet Allocation (LDA) or Structural Topic Models (STM) [Blei et al., 2003, Roberts et al., 2014]. While those methods discover latent global structures unsupervised, we assume a pre-defined outcome label f^* determined by the researcher. Furthermore, while our results induce a monoid congruence on strings when assumptions hold globally (discussed in Section G), we deliberately avoid assuming a global algebra. Instead, we rely on "local" consistency: checks enforced only on the specific leaves and merges realized by the scheduler. This allows us to estimate the fidelity of a pipeline without requiring the oracle space $\mathcal{Y}$ to possess a strict algebraic structure that the task semantics may not justify. The tradeoff is scope: we restrict attention to decomposable coding, extraction, counting, or guided-summarization tasks where f^* is defined on bounded spans.

In short, we borrow the algebraic intuition of mergeable sketches, adopt the sufficiency criteria of decision theory, and insist on edge-wise, testable equalities where practitioners actually operate. With this context in place, we turn to the formal setup.
This appendix provides complete technical foundations, detailed proofs, and practical guidance for the hierarchical summarization framework. It is organized to support both theoretical understanding and empirical deployment.

Appendix A: Notation Recap and Local Foundations collects all notation, definitions, and the foundational Lemma 1 used throughout. Readers comfortable with the main text may skip to Appendix B.

Appendix B: Proofs of Main Results establishes schedule invariance (Corollary 1), fold-of-folds invariance (Corollary 2), and multi-round preservation (Theorem 2)—the three core guarantees that make hierarchical processing practical.
Appendix C: Blackwell-Style Score Transport shows how supervision signals factor through the oracle f^* under Assumption 1, enabling training on summaries instead of full documents.
Appendix D: Practical Audit Guidance provides step-by-step audit procedures for deployed hierarchies, including sample-size calculations and methods for composing leaf-level and merge-level distortions into deployment-level bounds.
Appendix E: Local Consistency Bootstrap describes the DSPy-style optimization loop that alternates between sampling local checks and updating the prompt parameters until the empirical violation rates fall below a desired threshold.
Appendix F: Direct Preference Optimization connects oracle preservation to preference learning, proving that DPO on summaries recovers the same optimal policies as DPO on full documents when the sufficiency, merge-consistency, and idempotence conditions hold.
Appendix G: Global Compatibility Variant presents an alternative formulation using global merge-consistency/associativity assumptions, connecting the framework to algebraic structures like monoid congruences and mergeable sketches.
Appendix H: Information-Theoretic View (optional) sketches how OPS can be interpreted through an information-decomposition lens akin to the Information Sieve [Steeg and Galstyan, 2016], though this perspective is not required for the main arguments.
Appendix J: Existence of Oracle-Preserving Summaries establishes that oracle-preserving summarizers always exist (set-theoretically) and can be implemented as compact encodings when the oracle space admits short representations.
Appendix K: Worked Examples instantiates the framework for binary event detection, entity extraction, counting aggregates, and provides a counterexample illustrating why stability cannot be omitted.

A Notation Recap and Local Foundations

This appendix collects the primitives needed before the first theorem in Section **??**. It spells out every object once, keeping the notation flexible enough that we can later drop explicit "g" symbols and simply write g for the hierarchical evaluation.

A.1 Strings, oracle, and metric

We work on the free monoid $(\mathcal{X}, \oplus, \varnothing)$: $\mathcal{X}$ is the set of all finite token sequences, $\oplus$ is the usual concatenation of strings, and $\varnothing$ is the empty sequence acting as the identity. Documents are random variables $X \in \mathcal{X}$ drawn from a population law ρ. When we fix a specific document, we denote its base, unsummarized form by x_0 so that the realized leaf blocks are literally contiguous substrings of x_0. A fixed oracle $f^* : \mathcal{X} \to Y$ maps strings to their task value (labels, tuples, rewards, etc.). A measurable (pseudo)metric $d_{\mathcal{Y}} : Y \times Y \to [0, \infty)$ scores task disagreement, and we use it directly as $d_{\mathcal{Y}}(f^*(z), f^*(x))$.

A.2 Summarizers and expectations

A summarizer is a (possibly stochastic) Markov kernel $g : \mathcal{X} \rightsquigarrow \mathcal{X}$ that turns any input string into a string-valued random output. Whenever randomness is present we average over the seeds or call counts used by g; when g is deterministic the equalities below hold pointwise. To avoid notational clutter we simply write $\mathbb{E}[\cdot]$ for all expectations and suppress the implicit conditioning on the realized substrings.

A.3 Partitions, trees, and hierarchical evaluation

Fix a document $x = b_1 \oplus \cdots \oplus b_k$; equivalently $x_0 = x$ and $b_1, \ldots, b_k$ are contiguous substrings drawn from x_0. A emphrealized partition and tree consists of those blocks together with a full binary tree T whose leaves, read left to right, are $(b_1, \ldots, b_k)$. Leaves therefore contain no summaries—only raw text—and internal nodes contain summaries with two ordered children denoted u_L, u_R. Each node $u \in T$ carries two associated strings. The latent raw span is

$$S(u) = \begin{cases} b_i, & u \text{ is the } i\text{th leaf}, \\ S(u_L) \oplus S(u_R), & u \text{ internal with children } u_L, u_R, \end{cases}$$

and the stored summary that flows upward is

$$g(u) := \begin{cases} g\big(S(u)\big), & u \text{ leaf}, \\ g\big(g(u_L) \oplus g(u_R)\big), & u \text{ internal.} \end{cases}$$

Summary of symbols and expectations. The following table collects the core notation used throughout the appendix and main text.

Symbol	Meaning
x_0	Base (unsummarized) document; each leaf block b_i is a contiguous substring
b	A realized leaf block input to g
u	Internal node with children u_L, u_R; raw text $S(u) = S(u_L) \oplus S(u_R)$
r	Root of tree T
$g(T')$	Summary at root of subtree T'; equivalently $g(T') = g(r')$ for root r' of T'
$Z^{(R)}(x)$	Rth-round summary: $Z^{(1)}(x) = g(r)$, $Z^{(R+1)}(x) = g(Z^{(R)}(x))$
$\mathbb{E}$	Expectation over randomness in g (pointwise if g deterministic)

Table 2: Core notation for hierarchical summarization.

A.4 Consistency conditions

The guarantees in the main text follow from three statements checked *only* on the leaves and internal nodes that the realized tree touches. All three definitions coincide with their counterparts in Section **??**; we restate them here to make the appendix self-contained.

Definition 8 (Sufficiency condition (C1))**.** *For each realized leaf b, the single-block summary preserves the oracle in expectation:*

$$\mathbb{E}\big[d_{\mathcal{Y}}\big(f^*(g(b)), f^*(b)\big)\big] = 0.$$

This is Eq. (C1) *in the main text.*

Definition 9 (Merge-consistency condition (C2))**.** *For every pair of strings $u, v \in \mathcal{X}$,*

$$\mathbb{E}\Big[d_{\mathcal{Y}}\Big(f^*(u \oplus v),\ f^*\big(g(g(u) \oplus g(v))\big)\Big)\Big] = 0.$$

The triangle inequality together with Condition (C1) immediately implies $\mathbb{E}[d_{\mathcal{Y}}(f^*(g(u) \oplus g(v)),\ f^*(g(g(u) \oplus g(v))))] = 0$, which is the specific check enforced at realized internal nodes in the hierarchy (Eq. (C3)*).*

Definition 10 (Idempotence condition (C3))**.** *For any random variable Z supported on the range of g,*

$$\mathbb{E}\big[d_{\mathcal{Y}}\big(f^*(g(Z)), f^*(Z)\big) \,\big|\, Z\big] = 0,$$

ensuring that re-summarizing a summary leaves the oracle unchanged (Eq. (C2)*).*

The edge-wise two-route comparison

$$d_{\mathcal{Y}}\Big(f^*(v \oplus w), f^*\big(g(v \oplus w)\big)\Big) = 0$$

for realized child summaries (v, w) is an optional audit probe (Remark 1).

Remark 2 (Task-level congruence)**.** *Whenever the distortion is zero, we may declare $x \sim x'$ if $d_{\mathcal{Y}}(f^*(x), f^*(x')) = 0$. This equivalence relation is a congruence on the free monoid (Appendix G) and motivates the view that g should output canonical representatives of task-equivalent strings.*

Notation snapshot.

A.5 Nodewise preservation and the first theorem

Lemma 1 (Nodewise preservation under the consistency conditions)**.** *If Conditions* (C1) *and* (C3) *hold on all realized leaves and internal nodes of T, then for every node u,*

$$\mathbb{E}\Big[d_{\mathcal{Y}}\big(f^*(g(u)), f^*(S(u))\big)\Big] = 0.$$

Proof. If u is a leaf, $g(u) = g(S(u))$ and Condition (C1) yields the claim. If u is internal with children u_L, u_R, the induction hypothesis gives $g(u_L) \sim S(u_L)$ and $g(u_R) \sim S(u_R)$. Condition (C2) applied to the raw inputs $S(u_L), S(u_R)$ implies

$$S(u_L) \oplus S(u_R) \;\sim\; g\big(g(S(u_L)) \oplus g(S(u_R))\big).$$

Because $g(S(u_L)) = g(u_L)$ and $g(S(u_R)) = g(u_R)$ by definition, the right-hand side equals $g(g(u_L) \oplus g(u_R)) = g(u)$. Thus $g(u) \sim S(u_L) \oplus S(u_R) = S(u)$, completing the induction. □

Proof of Theorem 1. Apply Lemma 1 to the root r of T. The conditional expectation $\mathbb{E}_r$ coincides with $\mathbb{E}$ because every coin flip used anywhere in the reduction of T sits below r. Moreover $S(r) = x$, establishing the claim. □

This lemma–theorem pair captures everything needed before invoking more elaborate schedules (Corollary 1) or additional rounds. From this point forward we simply write $g(u)$ for the summary stored at node u.

B Proofs of Propositions and Theorems in the Main Text

This section collects the proofs of all corollaries and theorems stated in Section **??**. These results establish three key invariance properties that make hierarchical processing practical: (i) schedule invariance—the expected oracle is the same whether we use balanced trees or daisy chains; (ii) fold-of-folds invariance—we can nest hierarchical reductions arbitrarily; and (iii) multi-round preservation—repeatedly "normalizing" summaries does not drift the oracle when on-range stability holds. All three follow from the consistency conditions (C1)–(C3) via structural induction.

Throughout this section we fix a document $x = b_1 \oplus \cdots \oplus b_k \in \mathcal{X}$ together with a realized full binary reduction tree T on the leaves $(b_1, \ldots, b_k)$. For a node $u \in T$ we write $S(u)$ for the realized string at u and $g(u)$ for the corresponding hierarchical evaluation defined in Appendix A. Expectations with $\mathbb{E}$ average *only* over the internal randomness of g used in the computation of $g(u_L)$, $g(u_R)$, and the parent call at u, conditional on the realized subtree below u.

Every result below works solely with the task pseudometric $d_{\mathcal{Y}}$; equalities are always stated after applying f^*. Lemma 1 and Theorem 1 already establish nodewise and root preservation under the three consistency conditions. Here we spell out three consequences: schedule invariance (any binary parenthesization on a fixed partition yields the same expected oracle), invariance under "fold-of-folds" execution plans, and multi-round preservation provided g is stable on its own range.

Proof of Corollary 1 (Schedule invariance). Lemma 1 applies to *any* realized node set. Changing the parenthesization merely changes which internal nodes appear, so the lemma re-runs verbatim and Theorem 1 yields the same root equality. $\square$

Proof of Corollary 2 (Fold-of-folds invariance). Let $(F_1, \ldots, F_J)$ be a contiguous coarsening of the leaves. For each j, fix an arbitrary full binary within-fold tree T_j whose leaves, read left to right, concatenate to F_j. Let $\widehat{T}$ be any full binary over-fold tree whose leaves, read left to right, concatenate to $F_1 + \cdots + F_J$. Form the *composite tree* T_{comp} with root r by grafting $T_1, \ldots, T_J$ into $\widehat{T}$ at its J leaves. If (C1) holds on the realized leaves of T_{comp} and (C3) holds at every realized internal node of T_{comp}, then

$$\mathbb{E}\Big[d_{\mathcal{Y}}\big(f^*(g(r)),\ f^*(x)\big) \Big] \ =\ 0.$$

In particular, the expected oracle at the root is invariant to the choice of the within-fold parenthesizations and to the over-fold parenthesization, provided the realized leaves and internal nodes of T_{comp} satisfy (C1)–(C3). Moreover, if (C2) (on-range stability) also holds, then for the usual two-level "fold-of-folds" computation that first reduces each F_j to $S_j := g(T_j)$ and then reduces $(S_1, \ldots, S_J)$ along $\widehat{T}$, we have

$$\mathbb{E}\Big[d_{\mathcal{Y}}\big(f^*(g(\widehat{T})),\ f^*(x)\big) \Big] \ =\ 0,$$

because all additional g calls at the leaves of $\widehat{T}$ are re-applications of g to on-range strings and are inert at the oracle by (C2). Build the composite tree T_{comp} that first runs $T_1, \ldots, T_J$ and then grafts their outputs into $\widehat{T}$. Every realized leaf or internal node across both levels appears exactly once inside T_{comp}, so Lemma 1 and Theorem 1 applied to T_{comp} give the displayed equality. For the two-stage implementation, note that the leaves of $\widehat{T}$ are already on the range of g, so on-range stability (C2) makes the extra applications of g inert in expectation. $\square$

Proof of Theorem 2 (Multi-round preservation). Set $\Delta_R(x) := \mathbb{E}[d_{\mathcal{Y}}(f^*(Z^{(R)}(x)), f^*(x))]$. The base case $\Delta_1(x) = 0$ is Theorem 1. For the inductive step, use stability on the range of g to note that $\mathbb{E}[d_{\mathcal{Y}}(f^*(g(Z^{(R)}(x))), f^*(Z^{(R)}(x)))] = 0$. The triangle inequality thus gives $\Delta_{R+1}(x) \le \Delta_R(x)$, and the sequence remains zero. $\square$

C Blackwell–Style Score Transport

When the task involves learning from supervision—scores, rewards, or preferences—hierarchical summarization must preserve not just the oracle $f^*(X)$ but entire conditional score processes. This section establishes *score transport*: under Assumption 1, any supervision signal that factors through f^* can be transported from documents to summaries without information loss. The key insight is that sufficiency in the Blackwell sense (the summary's σ-field contains the oracle's) implies equality of all linearly aggregated expectations, including proper scoring rules and linear utilities. For nonlinear objectives such as logistic transforms in preference learning, exact preservation still holds when the three consistency conditions are satisfied exactly; approximate preservation is controlled by the oracle distortion Δ_R quantified in Appendix D.

We recall Assumption 1: there exists a measurable $\bar{Q} : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}$ such that $\mathbb{E}[S^\star(X, a) \mid X] = \bar{Q}(f^*(X), a)$ for all $a \in \mathcal{A}$. The random object we compare across representations is always *real-valued*: either the conditional score process $a \mapsto \mathbb{E}[S^\star(X, a) \mid \cdot]$ or its linear images under bounded linear functionals Λ; we never take expectations of Y-valued quantities.

Exact score transport

The first statement is a Blackwell/Doob–Dynkin equality specialized to our setup. To avoid undefined expressions like $S^\star(Z,a)$, we define the summary-indexed score process by conditional expectation:

$$S^\star_Z(Z,a) \ := \ \mathbb{E}\big[S^\star(X,a) \mid Z\big], \qquad a \in \mathcal{A}.$$

This averages the population score over all documents X that compress to the same summary Z. Intuitively, if the summary preserves all oracle information, then the score process on summaries must match the score process on documents.

We now show that when the summary is sufficient for the oracle—meaning $\sigma(f^*(X)) \subseteq \sigma(Z)$, i.e., the oracle can be recovered from Z—all expected scores are identical whether computed on full documents or on summaries.

Lemma 2 (Blackwell/Doob–Dynkin score transport)**.** *Assume Assumption 1. Let Z be any representation with $\sigma(f^*(X)) \subseteq \sigma(Z)$ (i.e., Z is sufficient for $f^*(X)$). Then for every $a \in \mathcal{A}$, the expected score is the same whether computed on documents or summaries:*

$$\mathbb{E}\big[S^\star(X,a)\big] \ = \ \mathbb{E}\big[S^\star_Z(Z,a)\big] \ = \ \mathbb{E}\big[\bar{Q}(f^*(X),a)\big].$$

Moreover, for any bounded linear functional Λ on the Banach space $(\mathcal{B}(\mathcal{A}), \|\cdot\|_\infty)$ of bounded real functions on $\mathcal{A}$,

$$\mathbb{E}\big[\Lambda\big(S^\star(X,\cdot)\big)\big] \ = \ \mathbb{E}\big[\Lambda\big(S^\star_Z(Z,\cdot)\big)\big] \ = \ \mathbb{E}\big[\Lambda\big(\bar{Q}(f^*(X),\cdot)\big)\big].$$

Proof. By the law of iterated expectations, $\mathbb{E}[S^\star(X,a)] = \mathbb{E}\{\mathbb{E}[S^\star(X,a) \mid X]\} = \mathbb{E}[\bar{Q}(f^*(X),a)]$. Also $\mathbb{E}[S^\star_Z(Z,a)] = \mathbb{E}\{\mathbb{E}[S^\star(X,a) \mid Z]\} = \mathbb{E}[S^\star(X,a)]$. Applying the bounded linear functional Λ and using linearity completes the proof. □

A particularly transparent special case is when Z *preserves the oracle exactly*, i.e., there exists a measurable h with $f^*(X) = h(Z)$ almost surely (Doob–Dynkin). Then $S^\star_Z(Z,a) = \mathbb{E}[S^\star(X,a) \mid Z] = \bar{Q}(h(Z),a)$, so for any bounded Λ,

$$\mathbb{E}\big[\Lambda\big(S^\star(X,\cdot)\big)\big] \ = \ \mathbb{E}\big[\Lambda\big(\bar{Q}(h(Z),\cdot)\big)\big].$$

In particular, when $Z = Z^{(R)}(X)$ is the R-round hierarchical reduction and the three conditions hold *exactly* along the realized tree so that $f^*(Z^{(R)}(X)) = f^*(X)$ almost surely (Theorems 1 and 2), we may take $h(Z^{(R)}(X)) = f^*(Z^{(R)}(X))$ and obtain equality of all linearly aggregated supervision risks when training on X versus on $Z^{(R)}(X)$.

D Practical audit guidance

What matters at deployment is a single number,

$$\Delta_R \ := \ \mathbb{E}\Big[\, d_{\mathcal{Y}}\big(f^*(Z^{(R)}(X)),\ f^*(X)\big)\Big],$$

the expected *task–space* distortion after one hierarchical pass and $R-1$ optional re-summarization rounds. All statements below assume $d_{\mathcal{Y}} \in [0,1]$ (rescale if needed by $\operatorname{diam}(Y)$). The audit is local, deployment-specific, and piggybacks exactly on the two-route merge diagram: a parent can be reached either by "concatenate–summarize" or by "summarize children–concatenate–summarize again" (Figure 3).

Exact pass short-circuit. If the realized leaves satisfy (C1) exactly and the realized internal nodes satisfy (C2) exactly, Theorem 1 gives $\Delta_1 = 0$. If, in addition, (C3) holds, then Theorem 2 yields $\Delta_R = 0$ for all $R \geq 2$. In this regime there is nothing to bound; the audit reduces to checking that the empirical versions of the three quantities below are (within tolerance) zero.

What to estimate (local violation rates). Let a realized reduction tree have N leaves and $M = N - 1$ internal nodes. Denote realized leaves by b (each b is a contiguous substring of the base document x_0 and is never itself a summary), and realized internal nodes by u with children u_L, u_R whose stored summaries are $g(u_L), g(u_R)$. Define the *indicator* violations

$$\begin{aligned}
p_{\text{suff}} &:= \mathbb{E}\,\mathbf{1}\big\{d_{\mathcal{Y}}\big(f^*(g(B)),\, f^*(B)\big) > 0\big\}\,,\\
p_{\text{bound}} &:= \mathbb{E}\,\mathbf{1}\big\{d_{\mathcal{Y}}\big(f^*(g(B_i \oplus B_{i+1})),\, f^*(g(g(B_i) \oplus g(B_{i+1})))\big) > 0\big\}\,,\\
p_{\text{merge}} &:= \mathbb{E}\,\mathbf{1}\big\{d_{\mathcal{Y}}\big(f^*(S_L \oplus S_R),\, f^*(g(S_L \oplus S_R))\big) > 0\big\}\,,\\
p_{\text{idem}} &:= \mathbb{E}\,\mathbf{1}\Big\{d_{\mathcal{Y}}\big(f^*(g(Z^{(1)}(x))),\, f^*(Z^{(1)}(x))\big) > 0\Big\} \qquad \text{(only if } R \geq 2\text{)}.
\end{aligned}$$

Let $\lambda \in [0,1]$ denote the probability that a merge-consistency sample targets a leaf boundary (so $1 - \lambda$ is the probability it targets an internal merge). The aggregate violation rate used by the bounds below is

$$p_{\text{assoc}} := \lambda\, p_{\text{bound}} + (1-\lambda)\, p_{\text{merge}}.$$

The two components correspond exactly to the two links in Condition (C3): p_{bound} measures the joint-vs-disjoint comparison when raw concatenations fit in context, while p_{merge} measures the raw-vs-summary comparison on stored child strings. In code: log each raw leaf b (the substring of x_0 that entered the hierarchy), its summary $g(b)$, and, for every internal node, the ordered pair of stored child strings $\big(g(u_L), g(u_R)\big)$ together with the resulting parent summary $g(u)$. Also record the seeds/call counts used by g so runs can be replayed. Then
emphsample leaves and internal nodes,
emphreplay the same randomness policy when needed, evaluate the displayed $d_{\mathcal{Y}}$-terms via f^*, threshold at > 0, and average. Conditioning on the realized subtree in the merge term matches (C3) exactly.

Crude but rigorous aggregation by union bound. Define the event that the root deviates after R rounds by

$$\mathcal{E}_R := \Big\{\ d_{\mathcal{Y}}\big(f^*(Z^{(R)}(X)),\ f^*(X)\big) > 0\ \Big\}.$$

If no leaf fails Condition (C1) and no realized internal node fails Condition (C3) (and, for $R \geq 2$, no on-range re-summarization fails Condition (C2) at the root between rounds), then Theorems 1 and 2 force $\mathcal{E}_R$ not to occur. Therefore

$$\begin{aligned}
\mathcal{E}_R \subseteq & \bigcup_{\text{leaves } b} \{d_{\mathcal{Y}}(f^*(g(b)), f^*(b)) > 0\}\\
& \cup \bigcup_{\text{internal } u} \Big\{d_{\mathcal{Y}}\Big(f^*\big(g(g(u_L){+}g(u_R)\big),\ f^*\big(g(u_L){+}g(u_R)\big)\Big) > 0\Big\}\\
& \cup \bigcup_{t=1}^{R-1} \Big\{d_{\mathcal{Y}}\big(f^*(g(Z^{(t)}(X))), f^*(Z^{(t)}(X))\big) > 0\Big\}.
\end{aligned}$$

Taking probabilities and applying the union bound gives the *crude envelope*

$$\begin{aligned}
\Pr(\mathcal{E}_1) &\leq N\, p_{\text{suff}} + M\, p_{\text{assoc}},\\
\Pr(\mathcal{E}_R) &\leq N\, p_{\text{suff}} + M\, p_{\text{assoc}} + (R-1)\, p_{\text{idem}} \quad (R \geq 2).
\end{aligned}$$

Since $d_{\mathcal{Y}} \in [0,1]$, we have $\Delta_R = \mathbb{E}[d_{\mathcal{Y}}(\cdot,\cdot)] \leq \Pr(\mathcal{E}_R)$, hence

$$\Delta_1 \ \leq\ N\, p_{\text{suff}} \ +\ M\, p_{\text{assoc}}, \qquad \Delta_R \ \leq\ N\, p_{\text{suff}} \ +\ M\, p_{\text{assoc}} \ +\ (R-1)\, p_{\text{idem}} \quad (R \geq 2). \tag{2}$$

These bounds hold with no assumption beyond the consistency conditions used by Theorems 1 and 2. They are intentionally coarse: any single local failure is allowed to account for a root deviation. When (C2) holds, $p_{\text{idem}} = 0$ and the multi-round term vanishes.

Plug-in estimates. In deployment we report the sample plug-ins

$$
\begin{aligned}
\widehat{p}_{\text{suff}} &:= \frac{1}{n_\ell}\sum_{i=1}^{n_\ell} \mathbf{1}\{d_{\mathcal{Y}}(f^*(g(b_i)), f^*(b_i)) > 0\}\,,\\
\widehat{p}_{\text{bound}} &:= \frac{1}{n_{\text{bound}}}\sum_{j=1}^{n_{\text{bound}}} \mathbf{1}\Big\{d_{\mathcal{Y}}\Big(f^*\big(g(b_j \oplus b_{j+1})\big),\ f^*\big(g(g(b_j) \oplus g(b_{j+1}))\big)\Big) > 0\Big\}\,,\\
\widehat{p}_{\text{merge}} &:= \frac{1}{n_{\text{merge}}}\sum_{j=1}^{n_{\text{merge}}} \mathbf{1}\Big\{d_{\mathcal{Y}}\Big(f^*\big(s_{L,j} \oplus s_{R,j}\big),\ f^*\big(g(s_{L,j} \oplus s_{R,j})\big)\Big) > 0\Big\}\,,
\end{aligned}
$$

and, if $R \geq 2$,

$$
\widehat{p}_{\text{idem}} := \frac{1}{n_r}\sum_{k=1}^{n_r} \mathbf{1}\Big\{d_{\mathcal{Y}}(f^*(g(Z^{(1)}(x_k))),\ f^*(Z^{(1)}(x_k))) > 0\Big\}\,.
$$

Setting $\widehat{\lambda} := \frac{n_{\text{bound}}}{n_{\text{bound}}+n_{\text{merge}}}$ yields

$$
\widehat{p}_{\text{assoc}} := \widehat{\lambda}\,\widehat{p}_{\text{bound}} + (1-\widehat{\lambda})\,\widehat{p}_{\text{merge}}.
$$

The *crude deployment-level bound* is then

$$
\begin{aligned}
\widehat{\Delta}_1^{\text{ub}} &:= N\,\widehat{p}_{\text{suff}} + M\,\widehat{p}_{\text{assoc}},\\
\widehat{\Delta}_R^{\text{ub}} &:= N\,\widehat{p}_{\text{suff}} + M\,\widehat{p}_{\text{assoc}} + (R-1)\,\widehat{p}_{\text{idem}} \quad (R \geq 2).
\end{aligned}
$$

truncated at 1 if desired. Sampling should be random over the realized leaves and realized internal nodes of the run; if worried about depth or heterogeneity, stratify by depth or by child summary length and then aggregate. Labeling is whatever f^* requires (scripted or human).

Direct root estimation (often simplest). When you do not need a decomposition, estimate Δ_R directly by sampling documents x_i and computing $\widehat{\Delta}_R = \frac{1}{n}\sum_i d_{\mathcal{Y}}(f^*(Z^{(R)}(x_i)), f^*(x_i))$. If $R \geq 2$ and (C2) holds, $\Delta_R = \Delta_1$, so it suffices to measure $R = 1$.

E Local Consistency Bootstrap

This appendix instantiates the "consistency bootstrap" used throughout the main text. We view the hierarchical summarizer as a programmable policy $\mathcal{P}_\theta$ that exposes two callable modules:

- **LeafSummarizer**$(b; \theta_\ell)$ maps a raw block b to a summary s_b. Its loss is the indicator from Condition (C1).
- **MergeSummarizer**$((s_{u_L}, s_{u_R}); \theta_m)$ maps stored child summaries to a parent summary s_u. Its loss is the indicator from Condition (C3).

DSPy [Khattab et al., 2024] provides a natural implementation: each module is a DSPy "program" whose parameters are prompt slots, few-shot exemplars, and temperature controls. The bootstrap proceeds as follows.

1. Leaf pass. Sample a batch of leaves $\{b_i\}$. For each leaf, run **LeafSummarizer**, evaluate $d_{\mathcal{Y}}(f^*(g(b_i)), f^*(b_i))$, and store the trace $(b_i, s_{b_i}, \text{oracle})$. When the oracle disagrees, regenerate k candidates at elevated temperature or solicit a human rewrite. The first candidate that matches the oracle becomes a "positive" trace; if none exists, mark the edge as a hard negative for escalation. All traces are appended to DSPy's trace buffer and optionally distilled into few-shot exemplars.

2. Merge pass. Using the best available leaf summaries, sample adjacent pairs (s_{u_L}, s_{u_R}). Run **MergeSummarizer**, check $d_{\mathcal{Y}}(f^*(g(s_{u_L} \oplus s_{u_R})), f^*(s_{u_L} \oplus s_{u_R}))$, and record the trace. Successful merges are stored as exemplars. Failures trigger regeneration or escalation. Because the oracle only sees the concatenated child summaries, this step never requires reopening raw text.

3. Parameter update. Run DSPy's optimizer (e.g., coordinate descent over prompt slots or gradient-free search over temperatures) on the accumulated traces. The objective is simply the empirical violation rate: minimize the fraction of traces where the oracle disagrees. Both modules can be optimized jointly or in alternating rounds. When operated online, we interleave sampling and updates so that fresh violations immediately produce corrective exemplars.

4. Stopping rule. Maintain running estimates of $\hat{p}_{\text{suff}}$, $\hat{p}_{\text{assoc}}$, and (if applicable) $\hat{p}_{\text{idem}}$. Continue bootstrapping until these empirical rates fall below the deployment tolerance, as measured either on a dedicated validation set or via sequential confidence bounds. Hard negatives that never satisfy the oracle become the most valuable few-shot examples and should be preserved even after the rate target is met.

This loop requires no external supervision beyond the oracle assumed in the main body. Human edits or calls to a larger teacher model only appear when regeneration repeatedly fails, and those interventions are logged so that the improved spans become part of the next prompt. The resulting modules enforce the three consistency conditions by construction, aligning the entire hierarchy without ever inspecting the full document.

F Direct Preference Optimization (DPO): Equivalence and Gap Bounds

Direct preference optimization trains policies by comparing preferred ("win") vs. dispreferred ("loss") responses. This section shows when training on hierarchically summarized documents $Z^{(R)}(X)$ yields the same optimal policies as training on full documents X. Under exact oracle preservation—when Conditions (C1)–(C2) hold exactly along the realized tree—the DPO objective depends on the input only through $f^*(\cdot)$, so the sets of population minimizers coincide. When preservation is approximate, we control the population gap by the oracle distortion Δ_R and provide explicit bounds under Lipschitz or bounded-reward assumptions. The key requirement is that annotator preferences factor through the oracle; if preferences depend on presentational features not encoded by f^*, training on summaries can deviate even when oracle preservation holds.

Let preferences follow Bradley–Terry–Luce with per-oracle rewards $\{R_y : \mathcal{A} \to \mathbb{R}\}_{y \in \mathcal{Y}}$ and scale $\beta > 0$:

$$\Pr\{a^w \succ a^\ell \mid X = x\} = \sigma\Big(\beta\,[R_{f^*(x)}(a^w) - R_{f^*(x)}(a^\ell)]\Big), \qquad \sigma(t) = \frac{1}{1+e^{-t}}.$$

DPO reparameterizes the reward through the policy π relative to a reference policy π_{ref} and minimizes

$$\mathcal{L}_{\text{DPO}}(\pi; \pi_{\text{ref}}) = -\,\mathbb{E}_{(X,a^w,a^\ell)}\left[\log \sigma\Big(\beta\Big\{\log \frac{\pi(a^w \mid X)}{\pi_{\text{ref}}(a^w \mid X)} - \log \frac{\pi(a^\ell \mid X)}{\pi_{\text{ref}}(a^\ell \mid X)}\Big\}\Big)\right].$$

The log-ratio difference $\log \frac{\pi(a^w|X)}{\pi_{\text{ref}}(a^w|X)} - \log \frac{\pi(a^\ell|X)}{\pi_{\text{ref}}(a^\ell|X)}$ measures how much more the policy π favors the winner over the loser, relative to the reference. DPO trains π to maximize this gap for preferred pairs.

Definition 11 (Oracle-measurable policies)**.** *A policy π (and π_{ref}) is oracle-measurable if $\pi(\cdot \mid x) = \pi(\cdot \mid x')$ whenever $d_{\mathcal{Y}}\big(f^*(x), f^*(x')\big) = 0$.*

Theorem 3 (Exact DPO equivalence under oracle preservation and oracle–indexed pairs)**.** *Assume $d_{\mathcal{Y}}\big(f^*(Z^{(R)}(X)), f^*(X)\big) = 0$ almost surely for some $R \geq 1$. Assume also that the pair generator depends on X only through $f^*(X)$ and that π_{ref} is oracle-measurable with full support. Then the sets of population minimizers of $\mathcal{L}_{\text{DPO}}$ coincide when trained on X and when trained on $Z^{(R)}(X)$. Moreover, there exists a population minimizer that is oracle-measurable; equivalently, we may without loss of generality restrict attention to oracle-measurable policies.*

Proof. Write $Y = f^*(X)$ and note that exact preservation implies $f^*(Z^{(R)}(X)) = Y$ almost surely. Under the oracle-indexed pair generator and oracle-measurable π_{ref}, the joint law of (Y, a^w, a^ℓ) is the same whether the input to the learning objective is X or $Z^{(R)}(X)$, and the two inner logits depend on the input only through Y. Therefore $\mathcal{L}_{\text{DPO}}$ decomposes as an expectation over Y of a functional that depends on $\pi(\cdot \mid \cdot)$ only through its Y–sections. For each y, any Bayes–optimal $\pi(\cdot \mid y)$ solves an identical one-dimensional problem under X and under $Z^{(R)}(X)$, so the sets of minimizers coincide and every minimizer is Y–measurable (oracle-measurable). $\square$

Theorem 4 (Quantitative DPO gap under metric distortion)**.** *Let $\Delta_R = \mathbb{E}\big[\, d_{\mathcal{Y}}\big(f^*(Z^{(R)}(X)), f^*(X)\big)\,\big]$ be the expected oracle distortion. To bound the population gap quantitatively, we impose regularity conditions ensuring that policies do not react radically to small changes in the oracle value. One option is a* policy-Lipschitz log-ratio *condition: there exists $L_\pi < \infty$ such that*

$$\left| \log \frac{\pi(a \mid y)}{\pi_{\text{ref}}(a \mid y)} - \log \frac{\pi(a \mid y')}{\pi_{\text{ref}}(a \mid y')} \right| \leq L_\pi \, d_{\mathcal{Y}}(y, y') \quad \forall \, a, y, y',$$

so the policy's preference for action a changes smoothly with the oracle value. Alternatively, we may assume a reward-Lipschitz exponential family*: $\pi(a \mid y) \propto \pi_{\text{ref}}(a \mid y) \exp\{\beta^{-1} R_y(a)\}$ with $\sup_a |R_y(a) - R_{y'}(a)| \leq L_R \, d_{\mathcal{Y}}(y, y')$ for all y, y', which says the induced rewards vary smoothly in the oracle space. Then, writing $\mathcal{L}^X$ and $\mathcal{L}^Z$ for the DPO loss with inputs X and $Z^{(R)}(X)$, respectively,*

$$\big|\mathcal{L}^X(\pi) - \mathcal{L}^Z(\pi)\big| \leq \begin{cases} 2\,\beta\, L_\pi\, \Delta_R, & \textit{case (1)}, \\ 2\, L_R\, \Delta_R, & \textit{case (2)}, \end{cases} \qquad \textit{and} \qquad \inf_\pi \mathcal{L}^X(\pi) - \inf_\pi \mathcal{L}^Z(\pi) \leq \textit{same bound.}$$

Proof. Let $\varphi(t) = -\log \sigma(t)$, which is 1–Lipschitz. Denote the DPO logit by

$$\Lambda(X; a^w, a^\ell) := \beta \left[\log \frac{\pi(a^w \mid X)}{\pi_{\text{ref}}(a^w \mid X)} - \log \frac{\pi(a^\ell \mid X)}{\pi_{\text{ref}}(a^\ell \mid X)} \right].$$

Then

$$\big|\mathcal{L}^X(\pi) - \mathcal{L}^Z(\pi)\big| \leq \mathbb{E}\big|\varphi(\Lambda(X; a^w, a^\ell)) - \varphi(\Lambda(Z^{(R)}(X); a^w, a^\ell))\big| \leq \mathbb{E}\big|\Lambda(X; a^w, a^\ell) - \Lambda(Z^{(R)}(X); a^w, a^\ell)\big|.$$

In case (1), oracle-measurability lets us replace X by $y = f^*(X)$ and $Z^{(R)}(X)$ by $y' = f^*(Z^{(R)}(X))$, and the Lipschitz condition yields

$$\big|\Lambda(X; a^w, a^\ell) - \Lambda(Z^{(R)}(X); a^w, a^\ell)\big| \leq \beta\, L_\pi \big(d_{\mathcal{Y}}(y, y') + d_{\mathcal{Y}}(y, y')\big) = 2\,\beta\, L_\pi\, d_{\mathcal{Y}}(y, y').$$

Taking expectations gives the stated $2\,\beta\, L_\pi\, \Delta_R$ bound. In case (2), write the logit difference as $\beta^{-1}\big[(R_y(a^w) - R_{y'}(a^w)) - (R_y(a^\ell) - R_{y'}(a^\ell))\big]$ and apply the reward-Lipschitz bound twice to get $2\, L_R\, d_{\mathcal{Y}}(y, y')$; take expectations to conclude. $\square$

Remarks. (i) Because $d_{\mathcal{Y}} \geq 0$, any display of the form $\mathbb{E}[d_{\mathcal{Y}}(\cdot,\cdot)] = 0$ immediately implies $d_{\mathcal{Y}}(\cdot,\cdot) = 0$ almost surely with respect to the summarizer's randomness, conditional on the realized subtree. We use the stronger almost-sure formulation in the exact-equivalence statements for clarity. (ii) If π_{ref} is *not* oracle-measurable, exact preservation of f^* alone does not guarantee equivalence of minimizers; the KL term can transmit presentational artifacts of X into the optimizer. This is why the oracle-indexed pair generator and oracle-measurable π_{ref} hypothesis appears in Theorem 3.

G Global Compatibility Variant and Algebraic Structure

This section records a *global* counterpart to the local, auditable laws used in the main text. The point is conceptual: if the two-route identities were to hold *for all* pairs in $\mathcal{X} \times \mathcal{X}$, and if on-range parent calls depended only on child oracle values, then concatenation induces a *merge law on oracle values.* All equalities below are stated in the task metric $d_{\mathcal{Y}}$: when we write that two oracle outputs are "equal," this always means their $d_{\mathcal{Y}}$–distance is zero.

Remark 3 (How to read "equal at the oracle")**.** *We will say y and y' in $\mathcal{Y}$ are* equal at the oracle *if $d_{\mathcal{Y}}(y, y') = 0$. This can be thought of as identifying outcomes that the task metric cannot distinguish. More formally, this is the same as passing to the equivalence relation $y \sim y'$ iff $d_{\mathcal{Y}}(y, y') = 0$ and reading all statements modulo $\sim$.*

Global laws. We fix a *deterministic* summarizer $g : \mathcal{X} \to \mathcal{X}$ and assume three global properties.

Assumption 3 (Global f^*–sufficiency)**.** *For every $z \in \mathcal{X}$,*

$$d_{\mathcal{Y}}\big(f^*(g(z)),\ f^*(z)\big) = 0.$$

Assumption 4 (Global two-route compatibility)**.** *For every $u, v \in \mathcal{X}$,*

$$d_{\mathcal{Y}}\Big(f^*(u \oplus v),\ f^*\big(g\big(g(u) \oplus g(v)\big)\big)\Big) = 0.$$

Assumption 5 (On-range merge depends only on child oracles)**.** *There exists a measurable map $M : \mathcal{Y} \times \mathcal{Y} \to \mathcal{Y}$ such that for all $u, v \in \mathcal{X}$,*

$$d_{\mathcal{Y}}\Big(f^*\big(g\big(g(u) \oplus g(v)\big)\big),\ M\big(f^*(g(u)),\ f^*(g(v))\big)\Big) = 0,$$

and M is well-defined at the oracle level*: whenever $d_{\mathcal{Y}}(y_i, y_i') = 0$ for $i = 1, 2$, then*

$$d_{\mathcal{Y}}\big(M(y_1, y_2),\ M(y_1', y_2')\big) = 0.$$

The first law says a one-pass summary preserves the task value (globally, not just on realized leaves). The second is the global version of the edgewise two-route equality, with the *parent* summary explicit. The third law formalizes that, once inputs are on the range of g, the parent's oracle depends only on the two child oracle values; it also says this dependence respects "equality at the oracle" (outputs cannot change when inputs are changed within $d_{\mathcal{Y}}$–zero).

We now state two consequences.

Proposition 1 (Concatenation is a congruence for oracle equality on strings)**.** *If $a, a', b, b' \in \mathcal{X}$ satisfy $d_{\mathcal{Y}}\big(f^*(a), f^*(a')\big) = 0$ and $d_{\mathcal{Y}}\big(f^*(b), f^*(b')\big) = 0$, then*

$$d_{\mathcal{Y}}\big(f^*(a \oplus b),\ f^*(a' \oplus b')\big) = 0.$$

Proof. Let $U := g\big(g(a) \oplus g(b)\big)$ and $U' := g\big(g(a') \oplus g(b')\big)$. By two applications of the triangle inequality,

$$d_{\mathcal{Y}}\big(f^*(a\oplus b), f^*(a'\oplus b')\big) \le \underbrace{d_{\mathcal{Y}}\big(f^*(a \oplus b), f^*(U)\big)}_{\text{(I)}} + \underbrace{d_{\mathcal{Y}}\big(f^*(U), f^*(U')\big)}_{\text{(II)}} + \underbrace{d_{\mathcal{Y}}\big(f^*(U'), f^*(a' \oplus b')\big)}_{\text{(III)}}.$$

By Assumption 4, terms (I) and (III) are zero. For (II), Assumption 5 gives

$$d_{\mathcal{Y}}\Big(f^*(U),\, M\big(f^*(g(a)), f^*(g(b))\big)\Big) = 0, \qquad d_{\mathcal{Y}}\Big(f^*(U'),\, M\big(f^*(g(a')),\, f^*(g(b'))\big)\Big) = 0.$$

Assumption 3 and the premises yield

$$d_{\mathcal{Y}}\big(f^*(g(a)), f^*(g(a'))\big) \le d_{\mathcal{Y}}\big(f^*(g(a)), f^*(a)\big) + d_{\mathcal{Y}}\big(f^*(a), f^*(a')\big) + d_{\mathcal{Y}}\big(f^*(a'), f^*(g(a'))\big) = 0,$$

and analogously $d_{\mathcal{Y}}\big(f^*(g(b)), f^*(g(b'))\big) = 0$. The oracle-level well-definedness in Assumption 5 therefore implies

$$d_{\mathcal{Y}}\Big(M\big(f^*(g(a)), f^*(g(b))\big),\ M\big(f^*(g(a')),\, f^*(g(b'))\big)\Big) = 0,$$

which forces (II)$= 0$ by triangle inequality. Hence the whole right-hand side is zero. □

Proposition 2 (A merge law on oracle values (associative up to $d_{\mathcal{Y}}$))**.** *Define, for oracle values that actually arise on-range, the binary operation*

$$y_1 \ \odot\ y_2 \ :=\ M(y_1, y_2).$$

Then for all y_1, y_2, y_3 in the on-range set $f^*(g(\mathcal{X}))$, we have

$$d_{\mathcal{Y}}\Big((y_1 \odot y_2) \odot y_3,\ y_1 \odot (y_2 \odot y_3)\Big) = 0,$$

and $y_0 := f^*(g(\varnothing))$ is a two-sided identity in the same $d_{\mathcal{Y}}$–sense:

$$d_{\mathcal{Y}}(y_0 \odot y,\ y) = d_{\mathcal{Y}}(y \odot y_0,\ y) = 0 \quad \textit{for all on-range } y.$$

Consequently, concatenation on $\mathcal{X}$ descends, via f^*, to an associative merge on oracle values up to the task metric.

Proof. Pick $a, b, c \in \mathcal{X}$ with $y_1 = f^*(g(a))$, $y_2 = f^*(g(b))$, $y_3 = f^*(g(c))$. By Assumption 5,

$$(y_1 \odot y_2) \odot y_3 \ \text{ is oracle-equal to }\ f^*\big(g\big(g\big(g(a) \oplus g(b)\big) \ \oplus\ g(c)\big)\big),$$

and

$$y_1 \odot (y_2 \odot y_3) \ \text{ is oracle-equal to }\ f^*\big(g\big(g(a) \ \oplus\ g\big(g(b) \oplus g(c)\big)\big)\big).$$

Applying Assumption 4 twice yields

$$d_{\mathcal{Y}}\Big((y_1 \odot y_2) \odot y_3,\ f^*\big((a \oplus b) \oplus c\big)\Big) = 0, \qquad d_{\mathcal{Y}}\Big(y_1 \odot (y_2 \odot y_3),\ f^*\big(a \oplus (b \oplus c)\big)\Big) = 0.$$

Since concatenation on $\mathcal{X}$ is literally associative, $(a \oplus b) \oplus c = a \oplus (b \oplus c)$, hence the two targets of the displays are equal and therefore at $d_{\mathcal{Y}}$–distance zero from one another. The identity statements follow in the same way with $b = c = \varnothing$ and Assumptions 3–5. □

Remark 4 (Dropping the parent g at a merge)**.** *By Assumption 3, $d_{\mathcal{Y}}\big(f^*\big(g(g(u)\oplus g(v))\big),\, f^*\big(g(u)\oplus g(v)\big)\big) = 0$ for all u, v. Thus, in Assumption 4, one may replace the right-hand side by $f^*\big(g(u) \oplus g(v)\big)$ without changing any conclusion in the $d_{\mathcal{Y}}$–sense. We keep the explicit parent call to mirror the deployed pipeline.*

What this variant does *not* assume. We have not assumed that M is literally associative on *all* of $\mathcal{Y}$, nor that f^* embeds $\mathcal{X}$ into $\mathcal{Y}$ as a strict monoid. Propositions 1–2 assert the exact strength warranted by the applications: along on-range values produced by g, and with equalities read in the task metric, the two-route global laws make concatenation behave as an associative merge on oracle values. This global structure can be useful when the task semantics truly admit a merge law (e.g., counts); the main-text guarantees, however, require only the local, auditable laws.

H Information-Theoretic View of OPS

This appendix makes explicit the connection between Oracle-Preserving Summarization (OPS) and the *Information Sieve* of Steeg and Galstyan [2016]. The sieve decomposes an input X into a latent factor Y and *remainder information* $\bar{X}$ such that $(Y, \bar{X})$ reconstructs X exactly while Y captures as much multivariate dependence (Total Correlation) as possible and $\bar{X}$ is independent of Y. OPS instantiates the same architecture, but the objective is supervised: rather than maximizing dependence within X, we require that the retained summary $Z = g(X)$ be a sufficient statistic for the oracle $f^*(X)$.

Theorem 5 (Oracle information decomposition)**.** *Let X be a block or node in the hierarchy, let $Z = g(X)$ be its summary, and let ν denote any (possibly stochastic) remainder such that X is measurable with respect to (Z, ν). Then*

$$I(X; f^*) \;=\; I(Z; f^*) \;+\; I(\nu; f^* \mid Z).$$

In particular, Z preserves all oracle information if and only if $I(\nu; f^* \mid Z) = 0$.

Proof. Apply the chain rule for mutual information with $X = (Z, \nu)$. □

Condition (C1) (Sufficiency) enforces $I(\nu; f^* \mid Z) = 0$ at every realized leaf by checking that the oracle agrees on b and $g(b)$. Condition (C2) (Merge Consistency) ensures that when Z is formed by hierarchically merging child summaries (Z_{u_L}, Z_{u_R}), the remainder created at that merge is also orthogonal to f^*. Combining Theorem 5 with the incremental construction of the Information Sieve (Theorem 2.1 of Steeg and Galstyan [2016]) yields an additive view of the hierarchy: each merge acts as a sieve layer whose discarded bits are independent of the oracle. Consequently, if the local audits certify that $I(\nu; f^* \mid Z) = 0$ on every edge, the root summary preserves the full mutual information $I(X_{\text{root}}; f^*)$ present in the original document.

Finally, Section 5 of Steeg and Galstyan [2016] observes that discarding the remainder induces a lossy compression of the raw input. OPS inherits this interpretation: summaries $g(x)$ need not suffice to reconstruct the surface text, but they are *lossless with respect to f^**. The "loss" lives entirely in the nuisance remainder, which by design is independent of the oracle and therefore irrelevant for downstream coding tasks.

I Relationship to Mergeable Summaries

This appendix formalizes the connection between our Oracle-Preserving Summarization (OPS) framework and the theory of *mergeable summaries* developed for streaming and distributed databases [Agarwal et al., 2013a]. We show that OPS strictly generalizes mergeable summaries: when the oracle f^* is symmetric and the summarizer g is deterministic, our Parentwise Compatibility law (C3) is identical to the mergeability condition. When f^* is order-sensitive or g is stochastic, OPS extends the notion of mergeability to the free monoid of strings by replacing hand-designed merge operators with learned semantic merges implemented by an LLM.

Component	**OPS (this paper)**	**Mergeable summaries**
Input	Strings $u, v \in \mathcal{X}$	Datasets $D_1, D_2 \in \mathcal{D}$
Combination	Concatenation $\oplus$ (non-commutative)	Union $\uplus$ (commutative)
Summarizer	$g : \mathcal{X} \rightsquigarrow \mathcal{X}$	$S : \mathcal{D} \to \mathcal{S}$
Task/query	Oracle $f^* : \mathcal{X} \to \mathcal{Y}$	Query Q
Merge algorithm	$g(g(u) \oplus g(v))$	$\mathcal{M}(S(D_1), S(D_2))$
Consistency law	Parentwise compatibility (C3)	Mergeability condition

Table 3: Mapping OPS to the mergeable summaries framework.

I.1 Formal Setup of Mergeable Summaries

In the classical setting, data arrives as a collection of items, and the objective is to maintain a compact synopsis that can be merged efficiently. Let $\mathcal{D}$ denote datasets (typically multisets) and $\mathcal{S}$ the sketch space with $|\mathcal{S}| \ll |\mathcal{D}|$.

- **Data:** $D_1, D_2 \in \mathcal{D}$.
- **Operation:** Multiset union $\uplus$, which is associative and commutative.
- **Summary:** A function $S : \mathcal{D} \to \mathcal{S}$.
- **Query:** Q maps a sketch to the target statistic, with distortion bounded by ϵ.

Definition 12 (Mergeable Summary)**.** *A summary S is mergeable if there exists $\mathcal{M} : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ such that for all D_1, D_2,*

$$d_{\mathcal{Y}}(Q(\mathcal{M}(S(D_1), S(D_2)))\,,\, Q(D_1 \uplus D_2)) \le \epsilon.$$

The guarantee holds uniformly over arbitrary merge trees, allowing sketches to be composed without revisiting the raw inputs.

I.2 Mapping OPS to Mergeable Summaries

Table 3 aligns the notation used in Section 3 with the streaming literature. The distinction is that OPS works over the free monoid of strings $(\mathcal{X}, \oplus)$, where $\oplus$ is generally non-commutative, and uses an LLM summarizer g instead of a fixed algebraic operator.

I.3 Equivalence in the Commutative Case

Suppose the task ignores order, such as counting keyword occurrences or extracting the set of unique actors. In this case, f^* acts as a homomorphism from $(\mathcal{X}, \oplus)$ to a commutative monoid $(\mathcal{Y}, +)$:

$$f^*(u \oplus v) = f^*(u) + f^*(v),$$

up to the oracle metric $d_{\mathcal{Y}}$.

Parentwise compatibility then states that re-summarizing the child summaries is inert at the oracle:

$$\mathbb{E}\Big[d_{\mathcal{Y}}\Big(f^*\big(g(g(u) \oplus g(v))\big),\, f^*\big(g(u) \oplus g(v)\big)\Big)\Big] = 0.$$

When g is deterministic, the expectation drops and the condition becomes pointwise.

Proposition 3 (Reduction to Mergeable Sketches)**.** *Let f^* be commutative so that $f^*(u \oplus v) = f^*(v \oplus u)$, and let g be deterministic. If g satisfies exact Parentwise Compatibility* (C3) *on every realized merge, then g is a mergeable summary for f^* in the sense of Agarwal et al. [2013a].*

Proof. Define the merge algorithm $\mathcal{M}(s_1, s_2) = g(s_1 \oplus s_2)$, where $s_i = g(u_i)$ are summaries. For any u, v we compare two routes: (i) concatenate the summaries s_u, s_v and apply f^*, yielding $f^*(g(u) \oplus g(v))$; (ii) merge the summaries via $\mathcal{M}$ and apply f^*, yielding $f^*(g(g(u) \oplus g(v)))$. Parentwise compatibility enforces equality between these two values, so the mergeability condition holds with $\epsilon = 0$. When the oracle is commutative and Condition (C1) holds on the leaves, the concatenated summaries themselves agree with the concatenated raw text under f^*, recovering the classical guarantee. □

Note that, unlike classical sketches where data arrive already as sketch inputs, Condition (C2) in our framework also constrains the *raw* concatenations $u \oplus v$ to agree with their summarized counterparts. The parsing step from text to summary is therefore part of the algebra itself, which makes the OPS guarantees strictly stronger than standard mergeability claims.

Remark 5 (The universal merge)**.** *Classical sketches design bespoke $\mathcal{M}$ operators (vector addition for Count-Min, rank pruning for quantile sketches). OPS instead uses a universal merge: concatenate the summaries and apply the same summarizer again. The summarizer learns the algebra needed by the downstream oracle during its initial reduction pass, so no task-specific merge code is required.*

I.4 Generalization to Non-Commutative Tasks

The OPS framework extends mergeability to the free monoid where $u \oplus v \neq v \oplus u$. Narrative queries—"Did the protest in paragraph A trigger the crackdown in paragraph B?"—depend on order and context. Standard sketches over multisets cannot express these dependencies because the union operator discards order. Parentwise compatibility enforces a non-commutative analogue: the interaction between summarized blocks mimics the interaction between the original texts when evaluated by f^*. This is precisely the condition audited in Figure 3.

I.5 Error Propagation and Bounds

Mergeable summaries typically admit error bounds that grow logarithmically with tree height. Theorems 1 and 2 give an analogous "zero-error" propagation: if local merges satisfy Parentwise Compatibility exactly (in expectation), the global error collapses to zero regardless of tree shape. When local failures are bounded by ϵ, our deployment bound (1) plays the role of classical error accumulation: the total distortion is controlled by a weighted sum of leaf, merge, and stabilization failure rates. This is the semantic counterpart to bounding sketch errors with respect to the number of merge operations; it justifies the probabilistic audit in Section 4, where random sampling over the summary tree produces a high-confidence certificate for the entire hierarchy.

J Existence of Oracle-Preserving Summaries

Two existence notions matter: set-theoretic (*some* g exists) and operational (a *short* g exists that fits a budget). We state both in terms of the task metric.

Proposition 4 (Canonical representative map in metric form)**.** *Because $\mathcal{X}$ is countable, define $x \sim x'$ iff $d_{\mathcal{Y}}(f^*(x), f^*(x')) = 0$. Fix a total order on $\mathcal{X}$ and let $c : \mathcal{Y} \to \mathcal{X}$ send y to the least x with $d_{\mathcal{Y}}(f^*(x), y) = 0$. Then*

$$g_{\text{can}}(x) := c\big(f^*(x)\big) \quad \textit{satisfies} \quad d_{\mathcal{Y}}\big(f^*(g_{\text{can}}(x)),\, f^*(x)\big) = 0 \;\; \textit{for all } x,$$

and g_{can} is idempotent on its range. Hence $g_{\text{can}} \in \mathcal{G}^\star$ in the sense of Definition 7.

Proposition 5 (Compact encodings)**.** *If there exists an injective encoder* enc : $\mathcal{Y} \to \mathcal{X}$ *of uniformly bounded length such that* $d_{\mathcal{Y}}(f^*(\text{enc}(y)), y) = 0$ *for all* $y \in \mathcal{Y}$*, then*

$$g_{\text{enc}}(x) := \text{enc}(f^*(x)) \quad \textit{obeys} \quad d_{\mathcal{Y}}\big(f^*(g_{\text{enc}}(x)),\, f^*(x)\big) = 0$$

and is idempotent on its range. Thus $g_{\text{enc}} \in \mathcal{G}^\star$ *and returns short strings.*

Proposition 4 certifies internal consistency: exact, idempotent, oracle-preserving normal forms always exist; Proposition 5 records the operational version used in practice (the summary *is* a compact encoding of the oracle).

K Worked Examples and Templates

This section provides instantiation templates for four common task types, showing how to check Conditions (C1)–(C2) in practice. Each example specifies the oracle space $\mathcal{Y}$, the metric $d_{\mathcal{Y}}$, and concrete audit procedures for leaf sufficiency, merge consistency, and on-range idempotence. The first three examples cover standard deployment scenarios (binary event detection, structured entity extraction, and numeric aggregation); the fourth is a minimal counterexample demonstrating that on-range stability (Condition (C2)) cannot be omitted without breaking multi-round preservation.

K.1 Binary event detection (QA/passage-level)

- $Y = \{0, 1\}$, $d_{\mathcal{Y}}(y, y') = \mathbf{1}\{y \neq y'\}$.
- Sufficiency check: ask g to emit `EVENT: YES/NO` for each block; verify $\mathbb{E}[\mathbf{1}\{f^*(g(b)) \neq f^*(b)\}] = 0$ within CI.
- Merge-consistency check: for realized u, compare $f^*(g(u_L) \oplus g(u_R))$ to $f^*(g(g(u_L) \oplus g(u_R)))$; failures indicate cross-block evidence interactions not preserved by g.

K.2 Entity extraction (tuple)

- $Y \subseteq \mathcal{A} \times \mathcal{ACT} \times \mathcal{T} \times \mathcal{P}$ with weighted Hamming distance on fields.
- Canonicalization: require one actor/action per line in fixed order; idempotence checks use the same schema to keep g on-range.

K.3 Counting (numeric aggregation)

- $Y = \mathbb{N}$, $d_{\mathcal{Y}}(y, y') = |y - y'|$.
- Global variant instantiates $y_1 \odot y_2 = y_1 \oplus y_2$ under exact f^*-sufficiency; edgewise audits reduce to additivity checks at realized merges.

K.4 Stability Is Substantive: A Minimal Counterexample

Let $\mathcal{X}$ contain special tokens POS, NEG, and let $f^*(\text{POS}) = 1$, $f^*(\text{NEG}) = 0$, with general $f^*(x) \in \{0, 1\}$. Define

$$g(x) = \begin{cases} \text{POS}, & f^*(x) = 1,\ x \notin \{\text{POS}, \text{NEG}\}, \\ \text{NEG}, & f^*(x) = 0,\ x \notin \{\text{POS}, \text{NEG}\}, \\ \text{NEG}, & x \in \{\text{POS}, \text{NEG}\}. \end{cases}$$

Then C1 holds on leaves never equal to {POS,NEG}, but g is not on-range stable since $f^*(g(\text{POS})) = 0 \neq 1 = f^*(\text{POS})$. Thus additional normalization rounds can flip the label; C2 is required to guarantee Theorem 2.