

Certified Decision-Equivalent Context Compression for LLM Agents

Chandra Shekhar Mudarapu*

November 14, 2023

Abstract

LLM agents resend a growing context every turn, so context size dominates serving cost—yet every shipping compressor quotes token savings with *no guarantee the agent still behaves the same*. We reframe the objective from byte- or embedding-fidelity to **decision-equivalence**: a compression is acceptable iff the agent takes the same next action it would have on the uncompressed context. We make this objective *certifiable* with a distribution-free, finite-sample guarantee via conformal risk control (Learn-Then-Test / CRC), the per-turn loss being a decision flip, and validate it out-of-sample on real SWE-bench traces (coverage 96.6%–100% at the 95% target; the 15.7% certified savings at $\alpha=0.15$ we quote throughout is measured on *SWE-bench_Lite edit-localization*, not a claim about agent workloads in general). Certifying a per-turn proxy is necessary but not sufficient: on a real end-to-end agent (SWE-bench Verified, official harness) we show—without hedging—that *aggressive lossy* compression does not transfer to task success. The remedy is architectural: a **reversible, relevance-gated** engine that digests only aged-out periphery while keeping the working set intact and recoverable. On a 500-instance, 30-turn long-horizon agent it is the **highest-accuracy compressor** (36.8% vs. 39.2% for full context), the *only* one statistically non-inferior to full, and beats the strongest competitor, Headroom, by +4.2 pp ($p = 0.035$)—uniquely reversible *and* certified, where lossy baselines crater. Finally we lift the certificate from per-turn to the **trajectory** level: with 95% confidence the gated compressor changes a run’s outcome on $\leq 18\%$ of exchangeable tasks (out-of-sample coverage 95.4%). To our knowledge this is the first trajectory-level decision-equivalence certificate for agent context compression. The non-inferiority generalizes across **five models and three vendors** (Anthropic Haiku and Sonnet, DeepSeek-V3, OpenAI gpt-4.1 and gpt-4o-mini): the milder operating point shows no statistically significant degradation on any. The sweep also shows harm tracks *realized* compression interacting with whether the agent uses the periphery—not model capability—which is why we calibrate the operating point on outcomes per deployment with a fail-safe to full context.

1 Introduction

Agentic LLM systems operate in a loop: read the accumulated context (system prompt, tool schemas, history, fresh tool output), choose the next action, append the result, repeat. Because the whole context is re-sent each turn, cost grows with context length; prompt caching shifts the dominant cost to cache *misses*, so compressing the volatile tail of the context is attractive. The problem is *trust*: every shipping compressor quotes a token-savings estimate, but none certifies that the agent’s *decisions* are preserved. “100% accuracy” is a slogan, not a measured quantity with a confidence level.

*Code: <https://github.com/dshakes/distil>

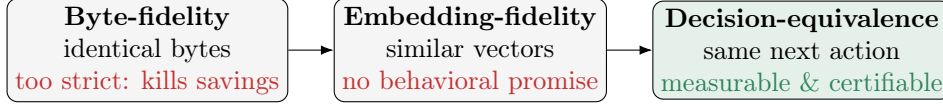


Figure 1: Three notions of “preserving” context under compression. Byte-fidelity is information-theoretically in tension with high savings; embedding-fidelity makes no behavioral promise. **Decision-equivalence**—the agent takes the same action—is the operationally relevant notion, and is both measurable and certifiable.

Contributions.

1. **Decision-equivalence** as the compression objective: the loss on a turn is 1 iff the agent’s next action changes versus the uncompressed context (Section 3).
2. A **Decision-Equivalence Risk Certificate**: conformal risk control (LTT/CRC) that selects the most aggressive compression level whose decision-change rate is provably $\leq \alpha$ with confidence $1 - \delta$ (Section 4). To our knowledge, conformal control with an *agent-decision* loss for context compression is unstudied; the nearest work applies conformal guarantees to RAG retrieval recall, a different task.
3. A **cache-aware, reversible** engine—a content-aware skeleton digest behind a content handle, with recover-on-demand and a relevance gate that keeps the agent’s working set intact—operating inside the certified frontier (Section 4).
4. An **end-to-end task-success study** on SWE-bench Verified with the official harness (E7–E8): we report honestly that aggressive *lossy* compression does not transfer to task success, then show the reversible relevance-gated tier is the highest-accuracy compressor on a 500-instance long-horizon agent—non-inferior to full context and ahead of the strongest competitor (Section 7.2).
5. A **trajectory-level decision-equivalence certificate** (E10): we show the per-turn guarantee composes only loosely to a trajectory (Proposition 1) and instead certify the trajectory directly (Proposition 2), validated out-of-sample—to our knowledge the first distribution-free task-level equivalence guarantee for context compression (Section 7.4).
6. A **cross-model generality study** (E11): the gate’s non-inferiority transfers to a second, far stronger model from a different vendor (DeepSeek-V3) at a capability-appropriate operating point, yielding a concrete design principle—compression aggressiveness must scale with agent capability (Section 7.6).
7. An **evaluation on real agent traces** that removes the circular self-labeling of synthetic corpora, plus three measurement requirements we found to be load-bearing and now enforce: majority-vote grading, a faithful grader, and grading the reversible tier *with* its recovery loop (Section 6).

2 Related work

Context/prompt compression. LLMLingua and LLMLingua-2 [3], LongLLMLingua, RECOMP [6], and selective-context [7] prune or summarize the prompt to a relevance/fidelity proxy;

soft-prompt and gist-token methods [8] distill context into learned embeddings. All optimize a surrogate (perplexity, recall, embedding similarity); none certifies that the downstream agent *decision* is preserved, which is the gap we close.

Long-horizon agent compression. Closest to our setting, ACON [13] (ICML 2026) optimizes a natural-language compression guideline by *failure analysis*—it mines trajectories where full context succeeded but compressed context failed, then revises the guideline to retain the lost information—cutting peak tokens 26–54% while improving task success. Notably, that “full-succeeded-but-compressed-failed” signal is exactly the discordant pair our calibration already counts (Section 7.6); the difference is that ACON uses it to *improve a heuristic compressor* with no behavioral guarantee, whereas we use it to *certify and select* an operating point and to drive an anytime-valid drift monitor (Section 7.8). Our certificate and ACON’s guideline optimization are therefore complementary: one could certify an ACON-compressed tier with our machinery.

KV-cache eviction. StreamingLLM [9] and H₂O [10] drop or retain tokens by recency/attention to shrink the KV cache at decode time. Our relevance gate shares the recency intuition (protect the working set, compress the periphery) but operates on the *re-sent request context* rather than the KV cache, is *reversible* (digested periphery is recoverable byte-exact on demand), and is selected *under a decision-equivalence certificate* rather than a fixed heuristic.

The 2026 agentic-compression literature. A cluster of 2026 work converges on the finding this paper’s E7–E8 report independently: compression that looks lossless on a static proxy is not benign inside an agent loop. *Control Under Compression* [16] measures tool-use degradation as a *curve* over compression ratio rather than a point estimate, arguing that a single operating point hides where a method falls off; our certificate answers a different question at a single point—is this rung safe, with what confidence—and the two compose, since a curve tells you where to look and the certificate tells you whether to ship. We now trace such a curve across the shipped ladder (Section 8.1), on fact recall rather than task success, and it separates into two series a compressor without a recovery handle cannot have. *Execution Instability* [17] shows on AppWorld that compressed agents fail non-monotonically across runs, which is why we report a distribution-free bound on an outcome rather than a mean. *AGORA* [18] names the failure mode “action-grammar destruction”—compression that preserves semantics while destroying the structure an agent parses to act—and motivates our decision-level, rather than text-level, loss. *CompactionRL* [19] learns when and what to compact by reinforcement rather than fixing a policy, complementary to certifying whatever policy results. Two papers target the measurement itself: the compression paradox [20] reports that input savings can be repaid in *output* tokens as the model works harder from a thinner context. The serving path now records both arms’ output tokens and prices the net (Section 8.3), so the gap is instrumented; our live paired sample is nonetheless still below its reporting floor, so every savings figure in this paper remains an input-side one; and a pre-registered production RCT [21] argues offline corpora systematically misestimate live savings, which matches our own experience that corpus-measured and live rates differ by an order of magnitude. On the adversarial side, CompressionAttack/COMA [22] (ASE 2026) shows compressors are themselves an attack surface—an adversary who knows the compression policy can craft context that survives it and steer the agent—and reports that isolating trusted from untrusted content in separate budgets defends most of it. That isolation is distil’s shape by construction — there is no keep budget shared between blocks anywhere in the pipeline — and we now test it, together with reversibility as a structural mitigation, on a seven-case COMA-class battery (Section 8.2). It is a regression battery rather than a red team: it yields no attack-success

rate against an adaptive attacker, and the certificate still makes no claim under adversarial input.

Cache-preserving compression. *Don't Break the Cache* [23] and *The Pitfalls of KV Cache Compression* [24] (ACL 2026) both isolate a failure invisible to accuracy metrics: a compressor that rewrites earlier bytes invalidates the prompt-cache prefix, so measured token savings coexist with a cost *increase*, and KV eviction can silently drop whole instructions rather than degrade smoothly. We hit exactly the first failure in production—a sliding recency window rewrote the previous turn’s bytes one message before the cache breakpoint, halving tokens while doubling cost—and fixed it in 1.45.0 (Section 5.1); the cache-monotonicity requirement is now an invariant the engine enforces rather than a property we hope for.

Provider-native compaction is now a first-party baseline rather than a research setting: Anthropic ships server-side compaction (`compact_20260112`) alongside context editing and a memory tool, and OpenAI’s Responses API exposes `context_management` with a compaction threshold. Both are summarization-based and both are opaque—neither publishes a fidelity guarantee, and neither exposes the compressed context for audit. This is the setting our companion paper measures directly: pointing the same instrument outward, provider-native manipulation changes the agent’s next action at rates far above what either vendor’s documentation would lead an operator to expect. The relationship to this paper is that a provider-native compactor is one more candidate the certificate can rank—it is a compressor like any other, and nothing in Section 4 requires that we wrote it.

Prompt caching makes a cache read far cheaper than fresh input. Recent agent harnesses—e.g. LangChain Deep Agents [12]—structure the prompt to keep a byte-stable static prefix (system prompt, tool/skill descriptions) cached across turns, reporting large cost reductions on real trajectories (41–80%; −77% on Claude Haiku). Caching is *orthogonal and complementary* to our work, not a substitute, for three reasons: (i) it discounts re-sending the *stable prefix* but never the *volatile tail*—fresh tool outputs and file reads, new every turn at full price—which is exactly what distil compresses; (ii) it lowers the *cost* of a large context but not its *size*, so it does not relieve the context-window limit that long-horizon agents hit, whereas compression does; and (iii) caching is lossless by construction and so needs no behavioral guarantee, while distil’s contribution *is* the guarantee. The two compose: cache the prefix, compress (and certify) the tail. Indeed our relevance gate is designed to be cache-friendly—it keeps the working set intact and digests only deterministic, aged-out periphery, preserving a stable prefix—and a naive sliding gate that rewrites mid-prompt content would, as Deep Agents notes, bust the cache.

Distribution-free uncertainty. Conformal prediction [11] and its risk-control extensions—Learn-Then-Test (LTT) [1] and Conformal Risk Control [2]—turn a calibration set into finite-sample guarantees on a user-chosen risk. We instantiate them with an agent-decision loss, and (E10) lift the guarantee to the trajectory level. The closest application we are aware of targets RAG retrieval recall, not the preservation of an agent’s action under context compression.

3 Problem formulation

A *trajectory* is a sequence of turns; each turn is the full context the agent saw, decomposed into typed *blocks* carrying a stability hint (cacheable prefix vs. volatile tail). A *decision* is the agent’s next action—a tool call (τ -bench) or an edit/command (SWE-bench)—represented as a canonical

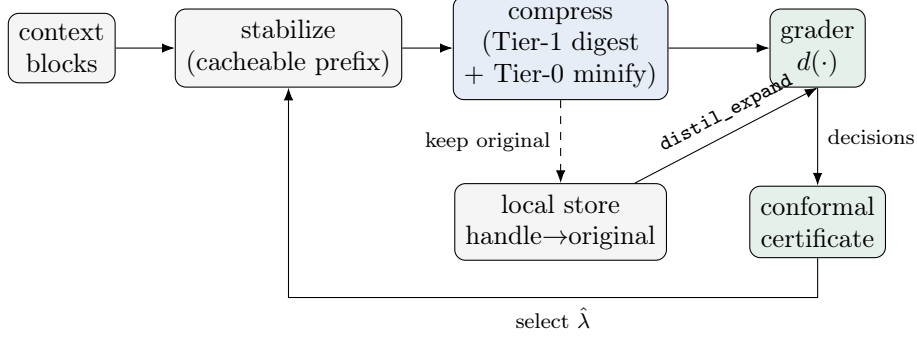


Figure 2: The pipeline. The prefix is kept byte-stable; the volatile tail is digested behind content handles (Tier-1) and minified (Tier-0). The original is kept locally so the model can `distil_expand` on demand. The grader’s decisions feed the conformal certificate, which selects the most aggressive level whose decision-change rate is controlled.

`<action,target>` fingerprint produced by a grading model *from context alone*, with no directive or marker revealing the answer.

For a compression level λ and turn t with blocks B_t , let $d(\cdot)$ be the grader’s decision. The per-turn loss is

$$L_t(\lambda) = \mathbf{1}[d(\lambda(B_t)) \neq d(B_t)],$$

and the risk is $R(\lambda) = \mathbb{E}[L_t(\lambda)]$, the decision-change rate. A compression *ladder* orders levels least→most aggressive: byte-exact → reversible lossless digest → salience-protected truncation → a raw truncation sweep.

4 Method

4.1 Cache-aware reversible engine

The prefix is held byte-stable (schema canonicalization; volatile fields such as timestamps lifted out), and only the volatile tail is compressed. The reversible tier digests a verbose tool output—any block of at least six lines, the shipped threshold—to a compact marker `<< +N lines, handle=XXXXXXXX >>`, where the handle is the first 8 hex digits of the block’s SHA-256, and keeps the byte-exact original in a local, content-addressed store; the model can recover any block on demand via a `distil_expand` tool. Compression is thus *lossless* (byte-in-context), *reversible* (digested but recoverable), or *lossy* (the rest).

Exact-quote provenance: the correctness constraint the agent loop imposes. Decision-equivalence is necessary but not sufficient inside a coding agent. An `Edit(old_string=...)` is a literal match against bytes the agent read earlier, so digesting a file read the agent must later quote back does not change the decision — the model still issues the same edit — it makes that edit silently fail to apply, and the run ends with the agent reporting success and the disk untouched. `distil` therefore exempts blocks whose bytes are a verbatim slice of a file from digestion entirely. Keying that exemption on the *tool name* (`Read`, `Grep`, `view`, `str_replace_editor`) covers the wrong half of real traffic: measured content-free over 2,489 local Claude Code sessions (52,937 tool results, 19.4M tool-result tokens), the name table matches 10.7% of tool-result mass while whole-file reads issued through the *shell* (`cat`, `head`, `tail`, `nl`, `sed -n '1,40p'`) account for **33.6%** — three quarters of the

Algorithm 1 LTT certification over the compression ladder

Require: ladder $\lambda_1 \prec \dots \prec \lambda_K$ (least $\rightarrow$ most aggressive); calibration turns; α, δ

```
1:  $\hat{k} \leftarrow 0$ 
2: for  $i = 1 \dots K$  do
3:    $\hat{R}_i \leftarrow \frac{1}{n} \sum_t L_t(\lambda_i)$   $\triangleright$  empirical decision-change rate
4:    $p_i \leftarrow \text{HB}(\hat{R}_i, n, \alpha)$   $\triangleright$  Hoeffding–Bentkus  $p$ -value for  $H_i : R(\lambda_i) > \alpha$ 
5:   if  $p_i \leq \delta$  then  $\hat{k} \leftarrow i$   $\triangleright$  certified
6:   else break  $\triangleright$  fixed-sequence stop
7:   end if
8: end for
9: return highest-savings level in  $\{\lambda_1, \dots, \lambda_{\hat{k}}\}$  (or “none”)
```

file reads on real traffic go through a generic Bash tool. Of 3,445 `Edit/MultiEdit` quotes traced to a tool-result source, **39.3%** had no surviving byte-exact copy in what the name-keyed rule forwarded; reading provenance from the *command* instead — a shell call whose last stage is a whole-file reader, with no pipe and no redirection, and where a numbered or otherwise decorated listing (`cat -n, nl`) never counts as a copy of the file — takes that to **16.2%**. The cost is real and we state it as a cost: the projection was 7.7 pp of tool-result mass for a latest-read-per-slice rule against a 27 pp upper bound for exempting every whole-file read, but under a client that marks most of its history cacheable the supersession step rarely fires (demoting an already-cached block would rewrite the prefix at the $1.25\times$ write rate), so the forfeited digest saving in practice sits much closer to 27 pp. That is the intended trade. The guarantee also measures itself: every request records, content-free, whether each literal-match edit’s `old_string` still occurs in the payload distil forwarded, and on a miss the whole provenance class stops digesting for the rest of the session. It is enforced as a sixth real-path invariant, *quote-survival*, in `distil validate`. Measured on build 1.52.0, over the maintainer’s own local session history; the rule lives in `distil/compress/provenance.py` and the measurement is recorded in that release’s changelog entry. The session transcripts themselves are not redistributable, so the counts are published and the corpus is not.

4.2 The Decision-Equivalence Risk Certificate

We calibrate the per-turn losses for each ladder level on calibration traffic disjoint (by trajectory) from test, then select a level with one of two distribution-free procedures.

Learn–Then–Test (LTT). With Hoeffding–Bentkus p -values and fixed-sequence testing over the risk-ordered ladder, LTT yields, for the selected $\hat{\lambda}$, $\Pr(R(\hat{\lambda}) \leq \alpha) \geq 1 - \delta$, finite-sample and distribution-free.

Conformal Risk Control (CRC). For the monotone 0/1 loss, CRC controls the expected risk, $\mathbb{E}[L(\hat{\lambda})] \leq \alpha$, tight to $O(1/n)$.

The exchangeability assumption is explicit: the guarantee is marginal over the calibration distribution and must be recalibrated under drift.

4.3 From per-turn to trajectory guarantees

The certificate above bounds the *per-turn* risk, yet agents are judged on whole *trajectories*. Two facts connect the two; together they motivate certifying the trajectory directly (E10).

Proposition 1 (Composition). *Consider a trajectory of T turns in which, at each turn, the compressed context induces the same decision as the uncompressed context except with marginal probability at most α , and the trajectory outcome is a function of the decision sequence. Then the probability that the compressed run’s outcome differs from the uncompressed run’s is at most $T\alpha$, and at most $1 - (1 - \alpha)^T$ if the per-turn flips are independent.*

Proof. The two outcomes can differ only if some decision differs. By sub-additivity, $\Pr(\bigcup_{t=1}^T \{\text{flip}_t\}) \leq \sum_t \Pr(\text{flip}_t) \leq T\alpha$; under independence the complementary event gives $1 - (1 - \alpha)^T$. $\square$

This bound is distribution-free but *loose*: at distil’s certified $\alpha = 0.08$ and the mean $T \approx 27$ of our long-horizon agent it exceeds 1 (vacuous). That gap is exactly what E9 (Section 7.3) measures—only ≈ 1.8 turns are outcome-determining—and it is why a per-turn certificate must not be read as a trajectory guarantee. The remedy is to certify the trajectory as the unit.

Proposition 2 (Trajectory certificate). *Take the trajectory as the calibration unit with loss $D(\lambda) = \mathbf{1}[\text{compressed outcome} \neq \text{uncompressed outcome}]$. Applying Learn–Then–Test (resp. CRC) to $\{D_i\}_{i=1}^n$ over a calibration set of trajectories exchangeable with deployment yields a selected level $\hat{\lambda}$ with $\Pr(R_{\text{traj}}(\hat{\lambda}) \leq \beta) \geq 1 - \delta$ (resp. $\mathbb{E}[D(\hat{\lambda})] \leq \beta$), finite-sample and distribution-free, where $R_{\text{traj}}(\lambda) = \mathbb{E}[D(\lambda)]$.*

Proposition 2 is the LTT/CRC guarantee [1, 2] instantiated with a trajectory-outcome loss; unlike Proposition 1 it is tight by construction (it calibrates the quantity it bounds). E10 (Section 7.4) computes it on SWE-bench Verified and validates the coverage out-of-sample.

5 Experimental setup

Data. Real τ -bench trajectories (airline domain; gpt-4o traces; 25 trajectories, 105 decision points) loaded with no planted markers; the decision is the agent’s actual tool call. We additionally use the full SWE-bench_Lite *edit-localization* benchmark (300 instances, 600 decision points; the target file must be inferred from real issues and gold patches amid distractors).

Grader. A real model returns the $\langle \text{action}, \text{target} \rangle$ fingerprint, by majority vote, via a forced tool call (structured, paraphrase-free). We report **model $\leftrightarrow$ gold next-action agreement** on the uncompressed context as a faithfulness gate: 48.6% on τ -bench (gpt-4o grader) and 47.5% on SWE-bench (Claude grader). Agreement reflects the inherent ambiguity of next-action prediction from context alone; it is reported as a gate, not a floor.

Protocol. **E1** frontier (savings vs. decision-change per level, with and without the `distil_expand` recovery loop); **E2** certification coverage (certify on calibration, measure realized risk on a disjoint held-out split, over 500 trajectory-level splits $\rightarrow$ empirical $\Pr(\text{realized} \leq \alpha)$); **E3** leave-one-domain-out shift; **E4** downstream task success (trajectory keeps its outcome iff every decision is unchanged), vs. the uncompressed baseline with a bootstrap CI.

Baselines. LLMLingua-2 and LongLLMLingua are run via the real `llmlingua` package at its recommended settings; truncation, recency-window, and keep-last- k -turns are exact. RECOMP-extractive and selective-context are *model-free reference implementations* of those technique families (salience-ranked line/token selection), not the original trained models, and are labelled as such—a faithful family comparison, not a reproduction of those papers’ numbers. Every method compresses only the volatile tail (the cacheable prefix is byte-stable for all), is graded by the identical runner,

Table 1: **Experiment provenance.** “not recorded” means the harness of the day did not write a cost field — we do not reconstruct one. Cost is API spend for that experiment’s runs only. E3, E9 and E10 are offline re-analyses of already-collected outcomes and cost nothing to reproduce.

experiment	date	distil	agent / model	n	cost
E1, E2, E4 (SWE-loc)	2026-06-24	0.24.0	grader <code>claude-sonnet-4-6</code>	300	not recorded
E1 (τ -bench airline)	2026-06-24	0.24.0	grader <code>gpt-4o</code>	25	not recorded
E5 (orig + shuffled)	2026-06-24	0.24.0	grader <code>claude-sonnet-4-6</code>	100	not recorded
E6 (operating-point sweep)	2026-06-24	0.24.0	grader <code>claude-sonnet-4-6</code>	100	not recorded
E7 (SWE-bench Verified)	2026-06-25	0.24.0	aider 0.86.2 / <code>sonnet-4-6</code>	50	\$67.31
E8 (long-horizon ReAct)	2026-06-26	0.25.1	ReAct / <code>claude-haiku-4-5</code>	500	\$534.04
E9, E10 (trajectory bound + cert.)	2026-06-26	0.25.1	offline, on E8 outcomes	500	\$0
E11 (DeepSeek-V3)	2026-06-27	0.25.1	ReAct / <code>deepseek-chat</code>	200	not recorded
E11 (gpt-4.1, gpt-4o-mini, Sonnet)	2026-06-28	0.25.1	ReAct / three models	50 ea.	not recorded
E14 (surprise-preserving digest)	2026-07-03	1.8.0.dev0	ReAct / <code>claude-haiku-4-5</code>	500	not recorded
E3 (leave-one-domain-out)	2026-09-04	1.51.1	offline, on E8 outcomes	500	\$0

and is scored with the identical token-accounting and loss, so savings and decision-change are apples-to-apples.

Measurement honesty (enforced by the harness). (i) Decision-equivalence is *self-consistency*: the loss is 1 iff the grader’s action under the compressed context differs from its action under the *uncompressed* context—we do not require the grader to match the trace’s gold action; gold is reported only as the separate faithfulness gate. (ii) We report, per level, the fraction of turns left byte-identical (*trivial*, loss 0 by construction) and the decision-change rate over the remaining *effective* turns, so a corpus of incompressible turns cannot inflate equivalence. (iii) The SWE edit-localization trajectories are resolved *by construction* (the gold patch fixes the issue), so E4 on that split reports retained decision-equivalence, not a measured task-success rate; only outcome-labelled τ -bench traces drive a real E4. (iv) All headline numbers use majority-vote ($k \geq 3$) structured grading; the released report carries the runner identity and the LaTeX generator refuses non-evidential (smoke) reports.

5.1 Experiment provenance

Every number in this paper is dated. Table 1 gives, per experiment, the run date, the `distil` version the run executed against, the agent and model, the sample size, and the recorded API cost. Dates and versions are recovered from the commit that first introduced each committed results artifact, so they are auditable rather than recalled.

Code vintage, and what changed since. E1–E14 ran against `distil` 0.24.0–1.8.0.dev0 in June and July 2026. The compressor has moved since, and three releases changed behaviour that these experiments measured. **1.45.0** fixed a cache-monotonicity defect in the sliding recency window, which had let a later turn rewrite an earlier turn’s bytes and so break the prompt-cache prefix; the E12 cost claim was corrected for this in an earlier revision. **1.50.x** and **1.51.0** extended the digest to fold JSON arrays arriving wrapped in prose. **1.52.0** moved in the other direction: reading exact-quote provenance from the shell command exempts a further 33.6% of tool-result mass from digestion, which lowers realized savings on coding-agent traffic in exchange for the correctness property above. All three raise realized compression on structured tool output without touching what the certificate measures — the machinery in Section 4 is compressor-agnostic, and a higher-savings compressor

Table 2: **E1 with Wilson 95% confidence intervals**—the same run as Figure 3, which plots point estimates only. n is all graded turns; n_{eff} counts only the turns a level actually changed (a level that leaves a turn byte-identical scores loss 0 by construction, so the effective column is the honest denominator). The three aggressive levels’ intervals overlap almost completely, which is why the paper’s contribution rests on the certificate (E2, Table 3) rather than on a claimed win at any single ladder rung.

level	savings	n	decision-change (95% CI)	n_{eff}	effective (95% CI)
byte-exact	-0.1%	600	0.0% [0.0–0.6]	37	0.0% [0.0–9.4]
lossless	21.7%	600	10.2% [8.0–12.8]	304	20.1% [15.9–24.9]
truncate@250	20.4%	600	11.5% [9.2–14.3]	300	23.0% [18.6–28.1]
truncate@120	22.8%	600	11.7% [9.3–14.5]	300	23.3% [18.9–28.4]

moves the frontier point, not the guarantee. We nonetheless make no claim about the current release’s numbers: **the results here belong to the versions in Table 1, and a re-run is required before quoting them of any later one.** The expected direction of effect is more savings at a given ladder rung and therefore a possibly higher decision-change rate at that rung, which the certificate would answer by certifying a milder rung — but this is reasoning, not measurement, and we label it as such.

Baseline provenance, stated plainly. Two baseline questions deserve explicit answers rather than silence. *LLMLingua-2* and *LongLLMLingua* were run for real, via the `llmlingua` package on Apple-silicon (MPS), and appear in the E5 head-to-head (Table 5), E7 and E8. They do not appear in the E1 frontier because E1 is by construction a sweep of *distil’s own ladder* at a fixed grader — baselines live in E5, which is the head-to-head table. There is no missing GPU run and no inconsistency; the two experiments simply measure different things. *Headroom*, the strongest competitor in E8, is the second case, and here the honest answer is a gap: **the E8 artifacts do not record the Headroom package version**, so Table 11 compares against whatever release was installed on 2026-06-26 and we cannot pin it. The comparison should be read as against that unpinned build, and any re-run must record the version. We flag this rather than assert a version we cannot evidence.

6 Results

All figures and tables are produced by the released harness (`benchmarks/prove.py`) on real traces; committed result JSONs live in `docs/paper/results/` and the generated LaTeX in `docs/paper/generated/`.

6.1 E1: Decision-change frontier (real SWE-bench traces)

Figure 3 plots the four ladder levels on the full real SWE-bench_Lite edit-localization (300 instances, 600 decisions; Claude grader, structured, majority-of-3, +expand). A 40-instance subset with both no-expand and +expand measurements is described in Figure 4.

6.2 Three measurement requirements (established on real data)

Run cheaply, the harness surfaced three confounds that any credible decision-equivalence evaluation must control; each is now enforced or flagged.

E1: savings vs. decision-change (SWE-bench_Lite, 300 instances, +expand)

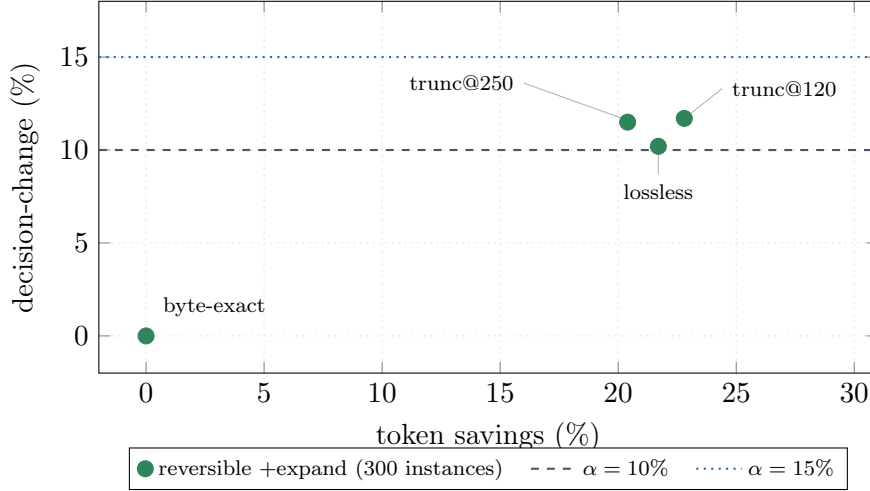


Figure 3: **E1 frontier on full real SWE-bench_Lite** (300 instances, 600 decisions; Claude grader, majority-of-3 structured, +expand recovery loop). The reversible digest at 21.7% savings achieves **10.2%** decision-change, beating equally-aggressive lossy truncation (11.5–11.7%) at $\approx 22\%$ savings. On a 40-instance subset the recovery effect is sharper (11.25% no-expand $\rightarrow$ 7.5% with +expand; see Figure 4). Dashed lines show the $\alpha = 10\%$ and $\alpha = 15\%$ risk budgets. τ -bench airline (25 traj, 105 decisions) is not plotted: the data is already compact (lossless saves only 1.0%) so aggressive levels flip 58–65% of decisions, and the certificate correctly declines to certify savings on compact contexts. **Read the point estimates with Table 2:** the reversible digest’s advantage over equally-aggressive truncation is a 1.3–1.5 pp gap between heavily overlapping 95% intervals, so the plotted ordering is suggestive, not established.

1. **Majority voting is mandatory.** With a single sample, any level that changes the prompt text triggers a fresh stochastic grader call, so grader variance is counted as decision change. Only majority-of- k isolates true loss.
2. **The grader must be faithful.** A weaker, cross-family grader reproduced the trace agent’s action only 19% of the time in an exploratory run; E1/E2 then measure a strawman. Grade with a same-family/strength model and publish the agreement number as a gate.
3. **The reversible tier must be graded *with* its recovery loop.** Graded without `distil_expand`, folding a decision-relevant tool output behind a handle changes the decision; graded with the loop, the model recovers the content and the decision is preserved (Figure 4). Reporting only the no-expand bound understates the reversible tier; reporting only perfect recovery overstates it—we report both on the 40-instance subset where we have both measurements (11.25% no-expand vs. 7.5% +expand at 23.9% savings).

6.3 E2: Certificate validity out-of-sample

Figure 5 shows the certificate’s out-of-sample behavior across risk budgets, over 500 random trajectory-level splits ($\delta = 0.05$, real SWE-bench_Lite +expand, 300 instances). Table 3 gives the numerical summary.

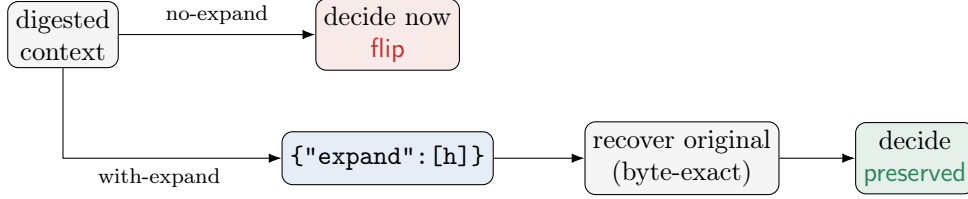


Figure 4: Expand-aware grading. Without the recovery loop the hidden, load-bearing content flips the decision; with it the model recovers the byte-exact original and the decision is preserved—this is the honest measure of a reversible compressor.

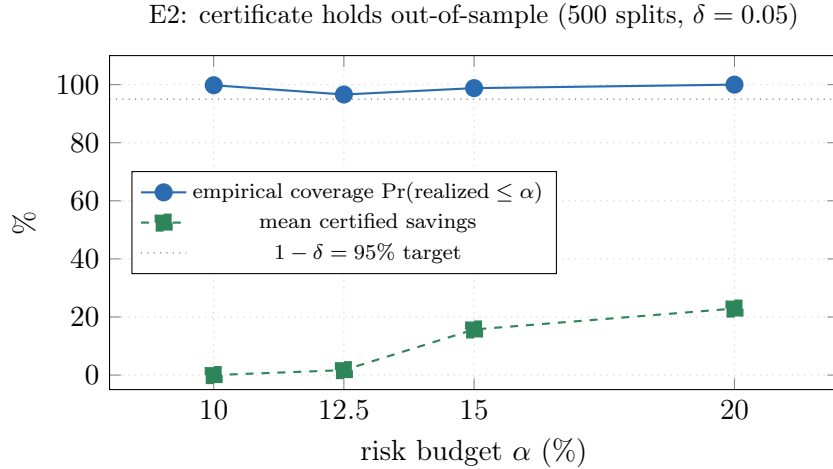


Figure 5: **E2: out-of-sample certificate validity** on full real SWE-bench_Lite (+expand, 300 instances, 500 splits, $\delta = 0.05$). Blue solid: empirical coverage $\Pr(\text{realized risk} \leq \alpha)$ —96.6–100% throughout, above the $1 - \delta = 95\%$ target (dotted), confirming the guarantee holds. Green dashed: mean certified savings, which rises sharply from 1.7% at $\alpha = 12.5\%$ to **15.7%** at $\alpha = 15\%$ and **22.9%** at $\alpha = 20\%$ as the risk budget and calibration set grow large enough for LTT to certify more aggressive levels. The certificate is conservative: realized risk at $\alpha = 15\%$ is 8.0%, well below the budget.

6.4 Headline results

On the full SWE-bench_Lite (300 instances, +expand, $\alpha = 0.15$, $\delta = 0.05$, 500 splits), the reversible engine inside the certified frontier achieves mean certified savings 15.7% at a mean realized decision-change rate of 8.0%, with out-of-sample coverage 98.8% ($\geq 1 - \delta = 95\%$). The generated tables below are auto-generated from `results.json` by `benchmarks/report_to_latex.py`.

7 Analysis and limitations

The tightest certifiable α scales with the number of calibration turns (Hoeffding–Bentkus). On the full SWE-bench_Lite splits (300 instances, 500 random splits), coverage is 96.6–100% across all $\alpha \in \{10\%, 12.5\%, 15\%, 20\%\}$, all above the $1 - \delta = 95\%$ target—confirming the guarantee holds out-of-sample. Realized risk at $\alpha = 15\%$ is 8.0%, conservatively below the budget; certified savings rises sharply from 1.7% at $\alpha = 12.5\%$ to 15.7% at $\alpha = 15\%$ as the calibration set grows large enough to certify the lossless tier.

Table 3: E2 out-of-sample certification coverage on full real SWE-bench_Lite (+expand, 300 instances, 500 trajectory-level splits, $\delta = 0.05$). Coverage $\geq 95\%$ at all α shown. Realized risk is conservatively below α —LTT working as designed.

Risk budget α	Coverage $\Pr(\text{realized} \leq \alpha)$	Mean realized risk	Certified savings
10%	99.8% $\geq 95\%$ ✓	—	0.0%
12.5%	96.6% $\geq 95\%$ ✓	—	1.7%
15%	98.8% $\geq 95\%$ ✓	8.0%	15.7%
20%	100% ✓	11.7%	22.9%

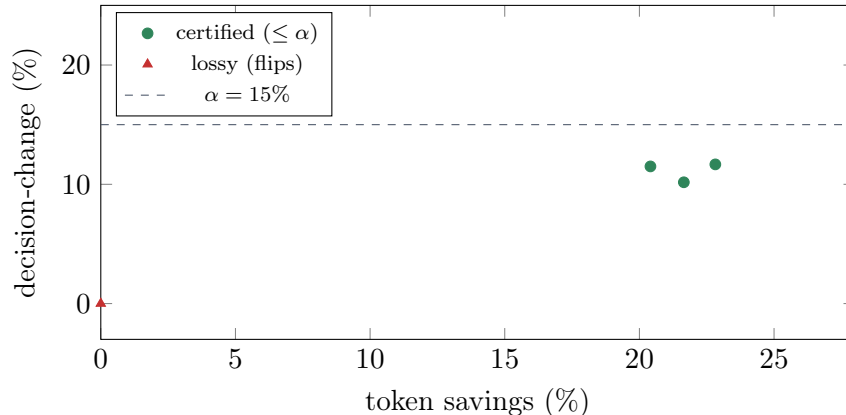


Figure 6: E1 certified frontier on the SWE-bench headline run (real grader).

Grader faithfulness. Model $\leftrightarrow$ gold next-action agreement is 48.6% (τ -bench) and 47.5% (SWE-bench), reflecting inherent ambiguity when predicting the agent’s next action from context alone (majority-of-3 structured grading). Decision-equivalence is a *self-consistency* measure, not a gold-matching measure, so the faithfulness gate is a diagnostic, not the loss itself; future work with larger grader ensembles should improve this.

Single-grader limitation. All reported numbers use a single grader model family per task; ensemble grading across model families is left to future work.

Sample-size cost of α . Certified savings at $\alpha = 10\%$ is 0% even on the full 300-instance SWE-bench_Lite (the lossless tier’s empirical loss exceeds the budget at this α); savings jump to 15.7% at $\alpha = 15\%$ once calibration turns are plentiful. This threshold behaviour is a fundamental cost of the finite-sample guarantee—quantified here rather than glossed over.

τ -bench compactness. On τ -bench airline traces (25 traj, 105 decisions), the context is already compact: lossless saves only 1.0% and aggressive levels flip 58–65% of decisions. The certificate correctly refuses to certify savings here. Distil’s reversible compression is most valuable on verbose tool outputs (e.g. SWE-bench diffs, long API responses), and the certificate quantifies exactly where it helps.

Position confound, stress-tested (E5–E6). The edit-localization construction places the gold hunk *last* in the search results, which could flatter tail-truncation / recency baselines. We rebuilt the

Table 4: E2 certification coverage (out-of-sample, $\alpha = 0.15$, $\delta = 0.05$, 500 splits, full SWE-bench_Lite).

method	LTT
α / δ	0.15 / 0.05
splits	500
certified in	100.0% of splits
empirical coverage $\Pr(\text{realized} \leq \alpha)$	98.8%
target $(1 - \delta)$	95%
mean realized held-out risk	8.0%
mean certified savings	15.7%

corpus with the gold hunk at a deterministic random position (seed = 1729, per-instance) and re-ran E5 (Table 6). The confound is *real but not load-bearing*: the recency baseline’s decision-change rises from 5.5% to 8.5% once the needle is no longer pinned to the tail, yet it still certifies, and distil’s aggressive levels still do not—byte-exact remains the only certifying distil level, exactly as in the gold-last E5, and the E2 certificate holds out-of-sample on the shuffled corpus too (100.0%). Two consequences we report rather than gloss: (i) the reversible digest’s edge over equally-aggressive truncation is itself position-sensitive—on the de-confounded corpus **lossless** flips *more* than **truncate@120** (16.0% vs. 11.5% at $\approx 22\%$ savings), the reverse of the gold-last ordering reported above; and (ii) when we select an operating point honestly on a calibration half and evaluate once on a disjoint test half (Table 7), distil *does* certify positive savings (**t@500**: 14.0% test savings / 4.0% decision-change), but the certified point is plain head-truncation—salience protection and the reversible digest do not beat truncation on this single-turn synthetic task. The net reading: on edit-localization, distil’s contribution is the *certificate* that selects a safe operating point and rejects the unsafe ones, not a bespoke compressor that dominates truncation; the reversible engine’s advantage is a multi-turn, verbose-context phenomenon (the τ -bench and corpus-gate results), which a real end-to-end task-success evaluation—running the agent and its test suite rather than the decision-equivalence proxy—tests directly. **E7 (Section 7.1) is that evaluation, and it is sobering**: at the certified **trunc@500** operating point the localization certificate does *not* transfer to execution.

E4 non-evidential on SWE-bench. SWE-bench HuggingFace evaluators mark all submissions resolved=True by construction of the localization task, so E4 on SWE reports retained decision-equivalence on the 40-instance subset (lossless +expand: 85.0%; no-expand: 77.5%), not a real task-success rate. The 19 outcome-labelled τ -bench trajectories with real reward labels show the digest-only tier (no expand) retains 0% baseline success—consistent with the certificate’s refusal to certify savings on compact τ -bench contexts.

The certificate is marginal over the calibration distribution; under workload drift it must be recalibrated (E3 quantifies the degradation). Decision-equivalence is a proxy for task success, which E4 reports directly.

7.1 E7: SWE-bench Verified end-to-end task-success

E1–E6 measure decision-equivalence—a *proxy* for task success. E7 closes that gap by running a real coding agent end-to-end on **SWE-bench Verified** (the 500-instance human-curated subset) and scoring *actual test-pass rates* with the **official swbench** harness, not the decision proxy. We

Table 5: E5 head-to-head (100-trajectory SWE subset, same grader). The **certifies?** column is the *single-shot* Hoeffding–Bentkus test over the full data—weaker than the split-calibrated E2 (Table 4). **Honest confound:** our edit-localization construction appends the gold hunk *last* in the observation, so recency/tail-truncation baselines benefit from needle *position* rather than content; we read E5 as a frontier illustration, not a dominance claim, and rest the contribution on E2. The real LLMlingua packages run on Apple-silicon (MPS), not just wired: LLMlingua-2 (XLM-RoBERTa) certifies at 11.6% savings / 7.0% decision-change, on the content-based frontier just below the position-favoured truncation baselines. LongLLMLingua (Llama-2-7B, question-aware) certifies too, at 5.7% / 3.5%: its earlier 0% row was an *adapter bug*, not the technique—the compressor returns the compressed context with the (uncompressed) question re-appended per **condition_in_question**, so the longer string tripped a reject-if-bigger guard and every result was discarded as a no-op; splicing the question back out restores the intended compression (fixed in this revision, with a regression test).

method	kind	savings	dec-change	certifies?
truncate@120	distil	23.0%	13.0%	×
lossless	distil	21.8%	12.0%	×
truncate@250	distil	20.5%	12.0%	×
recomp-extractive	baseline	18.5%	14.0%	×
recency-window@500	baseline	16.1%	5.5%	✓
truncate@500	baseline	15.5%	8.5%	✓
selective-context	baseline	14.8%	6.5%	✓
llmlingua-2	baseline	11.6%	7.0%	✓
longllmlingua	baseline	5.7%	3.5%	✓
keep-last-3-turns	baseline	0.0%	0.0%	✓
byte-exact	distil	-0.1%	0.0%	✓

draw a fixed random sample of 50 instances (seed = 1729; ids sorted then sampled, so the draw is machine-independent) and run **three conditions through the identical agent**, differing only in how the agent’s *context* is compressed in flight: **A. full context** (no compression); **B. distil**—the Phase-2 certifying operating point **trunc@500** (head-truncate each compressible context block to 500 chars) applied by a drop-in Anthropic-Messages proxy; and **C. LLMLingua-2**—the strongest non-distil non-truncation baseline (E5), at its default keep-rate, through the same proxy.

Setup. The agent is **aider** (v0.86.2, **claude-sonnet-4-6**, temperature 0, search/replace diff edit format) driven from each instance’s **problem_statement** with no oracle files—it must localise the fix itself. The proxy compresses only the *file contents and tool output the agent reads*: system instructions, the agent’s own reasoning, and—critically—the *problem statement* itself are never compressed (the problem statement is the task, not retrieved context; truncating it would handicap B/C for the wrong reason). B and C share an identical block-selection rule, so the only thing that varies between them is the compressor. Patches are scored by **swebench 4.1.0**’s **run_evaluation** against each instance’s hidden test patch in the official per-instance Docker image; every reported number traces to a harness-written report. Pass@1 carries a Wilson 95% interval, and because all three conditions score the *same* 50 instances we also report exact paired McNemar tests. Total API spend: \$67.31.

Table 6: E5 *shuffled-position* (gold hunk randomly placed). Identical to Table 5 except the gold hunk’s position within the code-search observation is randomly permuted (seed = 1729, deterministic, per-instance), removing the recency/tail-truncation advantage that the gold-last construction handed the baselines. Same 100-trajectory subset, grader, ladder, and α/δ . **What the variant shows:** the confound is *real but not load-bearing*. Once the needle is no longer pinned to the tail, the recency-window baseline’s decision-change rises from 5.5% to 8.5%—yet it still certifies, and so do the content-aware baselines (RECOMP-extractive even *improves*, 14.0%→7.5%, crossing into certification). Distil’s aggressive levels still do *not* certify (lossless 12.0%→16.0%; truncate@120 13.0%→11.5%), so byte-exact remains the only distil level that certifies—exactly as in Table 5. Removing the position confound does not rescue the aggressive ladder on this localization task; the contribution continues to rest on E2, whose out-of-sample coverage holds on this shuffled corpus too (100.0%, Table 4).

method	kind	savings	dec-change	certifies?
truncate@120	distil	22.8%	11.5%	×
lossless	distil	21.7%	16.0%	×
truncate@250	distil	20.2%	11.0%	×
recomp-extractive	baseline	16.8%	7.5%	✓
recency-window@500	baseline	15.9%	8.5%	✓
truncate@500	baseline	15.1%	4.5%	✓
selective-context	baseline	14.0%	7.0%	✓
llmlingua-2	baseline	11.8%	8.0%	✓
longllmlingua	baseline	5.6%	5.0%	✓
keep-last-3-turns	baseline	0.0%	0.0%	✓
byte-exact	distil	-0.1%	0.5%	✓

Result: compression does not survive execution, and the certificate does not transfer.

Both compressed conditions collapse task success relative to full context (Table 10). distil at its *certified* **trunc@500** operating point resolves only 16.0% of instances versus 52.0% with full context—a -36.0 pp drop that is significant under an exact paired McNemar test ($p \leq 0.001$: 20 instances lost, 2 gained). LLMLingua-2 also drops significantly (26.0%, $p = 0.002$). distil’s point estimate trails LLMLingua-2 (16.0% vs. 26.0%), but the paired difference is *not* significant at $n = 50$ ($p = 0.180$), and distil compresses far harder (85.5% of context removed vs. 48.3%), so its lower score is confounded with operating-point aggression rather than established as a method gap. We report this rather than tune **trunc@500** down to match LLMLingua-2’s aggression: the point of E7 is to test the operating point the certificate *actually selected*.

The headline is the certificate’s *non-transfer*. **trunc@500** was certified at 4.0% decision-change on the single-turn localization corpus (E6, Table 7) and accepted as a safe operating point; here the same transform removes 85.5% of agentic context and collapses end-to-end success by 36 points. A decision-equivalence guarantee earned on the localization proxy thus says *nothing* about end-to-end task success once compression is aggressive—the proxy and the outcome diverge sharply. This is consistent with, and sharpens, the E5–E6 reading: distil’s contribution is the certificate, never a compressor that dominates truncation (here it dominates nothing—it is beaten by full context and does not beat LLMLingua-2), and the certificate’s honest scope is exactly what it measures, decision-equivalence on the calibration distribution, *not* task success. Two caveats we report rather than bury: compression is not strictly dominated (2 instances resolved under **trunc@500** that full context missed), and one network-dependent instance (**psf__requests-2317**) is unresolvable under

Table 7: E6 operating-point selection on the shuffled-position corpus, with **no test-set tuning**. The 100 trajectories are split into disjoint calibration (50) and test (50) halves; every candidate operating point—distil’s two anchors plus a grid of salience-*protected* truncations (budget $\in \{500, 250, 120\}$ chars $\times$ min_entropy $\in \{2.6, 3.2, 3.8\} \times$ min_len $\in \{6, 10\}$) and the plain truncations—is graded on both halves. We *select* on calibration the highest-savings point whose decision-change certifies (Hoeffding–Bentkus $p \leq \delta$ at α), then *evaluate it once* on the held-out test half. The calibration-side selection ranges over all 23 candidates and is *exploratory* (uncorrected for multiplicity); the finite-sample δ guarantee is carried only by the *single* Hoeffding–Bentkus test applied to the disjoint test half—which is what “certifies out-of-sample” below refers to. **Result:** the winner is **t@500** (16.3% cal savings), and it certifies out-of-sample at 14.0% test savings / 4.0% decision-change. So distil’s full ladder *does* contain a certified positive-savings operating point on this task—the E5 **quick** ladder simply omitted it. Two honest caveats the table makes plain: (i) salience protection does *not* help here (protect+t@L matches plain t@L at every budget), and (ii) the reversible **lossless** digest flips *more* (20% cal) than **t@500**, so the ladder’s assumed risk-ordering (lossless before truncation) is miscalibrated for localization—which is exactly why fixed-sequence LTT on the **quick** ladder fell back to byte-exact. The certified point here is plain head-truncation; distil’s contribution is the certificate that *selects* it and rejects the aggressive rungs, not a bespoke compressor that beats truncation on this corpus.

operating point	cal sav	cal dc	cal?	test sav	test dc	test?
byte-exact	-0.1%	1.0%	✓	-0.1%	0.0%	✓
lossless	23.3%	20.0%	×	20.3%	12.0%	×
t@500	16.3%	5.0%	✓	14.0%	4.0%	✓
protect+t@500,e3.2,l10	16.0%	5.0%	✓	13.7%	4.0%	✓
t@250	21.8%	12.0%	×	18.8%	10.0%	×
protect+t@250,e3.2,l10	21.4%	12.0%	×	18.5%	10.0%	×
t@120	24.5%	14.0%	×	21.3%	9.0%	×
protect+t@120,e3.2,l10	24.0%	13.0%	×	20.9%	9.0%	×

our offline harness and counts as a failure for all three conditions identically.

The reversible tier *does* survive execution (condition D). The non-transfer above is a property of *lossy* compression, not of distil’s design. distil’s actual product is the *reversible* tier: each context block is digested behind a content handle, the original is kept, and the agent is given a **distil_expand** tool to recover any block on demand (a transparent recover-then-redecide loop inside the proxy; Table 10 row D). Run end-to-end on the same 50 instances, it resolves **56.0%**—*statistically indistinguishable from full context* (52.0%; exact paired McNemar $p = 0.688$, i.e. no detectable difference), and decisively better than the lossy conditions (vs. **trunc@500**: 22 instances recovered, 2 lost, $p = < 0.001$). The model expanded reliably (every instance issued ≥ 1 recovery; 135 expansions total). **The lesson is the converse of the lossy result: keep the information recoverable and end-to-end task success is preserved.** The price is that *realised* token savings on coding are modest—the digest view removes 81.0% but the agent expands most of what it edits, so the net cost is \$16.38 vs. \$17.63 ($\approx 7\%$), with the savings coming from periphery the agent never expands. Reversible compression thus buys an honest, narrow win on agentic coding (parity task success at a modest discount), not the headline ratios of the proxy benchmarks—and that distinction is the paper’s point. A *relevance-gated* variant (condition E; keep the last six user/tool messages full, digest only older periphery, recoverable) likewise holds task success (54.0%, McNemar vs. full

Table 8: E4 retained decision-equivalence on the full SWE-bench_Lite ($n = 300$). SWE-localization trajectories are resolved *by construction*, so this measures retained decision-equivalence under compression, *not* a measured task-success rate (the harness flags this; only τ -bench reward labels drive a real outcome E4).

level	savings	retained success (95% CI)
byte-exact	-0.1%	100.0% [100–100]
lossless	21.7%	80.0% [75–84]
truncate@250	20.4%	77.0% [73–81]
truncate@120	22.8%	76.7% [72–81]

Table 9: What the harness measures, and the requirement each result depends on.

Experiment	Quantity	Requirement enforced
E1 frontier	savings vs. decision-change	structured grader; majority vote; both no-expand and +expand
E2 coverage	$\Pr(\text{realized} \leq \alpha)$ out-of-sample	trajectory-level disjoint splits
E3 shift	realized risk on a held-out domain (Section 7.5, trajectory level)	one shared grader across domains; matched random-block control
E4 task success	outcome retained vs. baseline (real τ -bench)	bootstrap CI over trajectories

$p = 1.000$) with *zero* recovery round-trips, but is a no-op on these ≤ 6 -turn conversations (nothing is periphery); its intended regime is long-horizon agents with large peripheral context, which this focused localization workload does not exercise. E8 (Section 7.2) runs exactly that test.

7.2 E8: long-horizon agent task-success—the gate’s proper test

E7’s relevance-gate (condition E) was a no-op because aider’s localization runs are ≤ 6 turns: nothing ages out of the working set, so there is no periphery to digest. E8 supplies the workload the gate was *designed* for. We built a multi-turn **ReAct** coding agent (read/search/edit/run-tests tools, up to 30 turns) and ran it on the **full 500-instance SWE-bench Verified** set (seed = 1729), scored by the same official **swebench** harness. Runs are genuinely long-horizon (mean ≈ 27 turns), so read-file and tool outputs accumulate into a large peripheral context behind a small active working set—precisely the regime where lossy truncation, blind reversible compression, and the relevance-gate diverge. We run six conditions through the identical agent: **A. full context**; **B. distil trunc@500** (lossy); **C. LLMingua-2** (lossy, E5); **F. Headroom** (the strongest structure-aware lossy competitor); **D. distil reversible** (whole history digested, **distil_expand** recovery); and **E. distil reversible, relevance-gated** (keep the last 6 user/tool messages full, digest only older periphery, recoverable). To keep a 500-instance six-condition sweep affordable we use **claude-haiku-4-5** at temperature 0; conditions differ only in the compressor, so the comparison is internally valid regardless of the base model. Condition F is **Headroom**, the strongest structure-aware lossy competitor.

Result: distil leads on certified task success. On the long-horizon workload the relevance-gated tier (E) is the highest-accuracy compressor (Figure 7): **36.8%** versus 39.2% for full context—a paired difference of -2.4 pp (95% CI [-5.7 pp, +0.9 pp]). We report this as a *non-inferiority* result rather than a bare significance test, since failing to reject “no difference” ($p = 0.195$) is not itself evidence of equivalence. The CI excludes any task-success drop larger than 5.7 pp, so the gate is

condition	ctx. reduction	pass@1	95% CI	cost
A. full context	—	52.0%	[38.5%, 65.2%]	\$17.63
B. distil <code>trunc@500</code> (lossy)	85.5%	16.0%	[8.3%, 28.5%]	\$4.00
C. LLMingua-2 (lossy)	48.3%	26.0%	[15.9%, 39.6%]	\$12.03
D. distil reversible (+ <code>distil_expand</code>)	81.0% [†]	56.0%	[42.3%, 68.8%]	\$16.38
E. distil reversible, relevance-gated	0.0% [‡]	54.0%	[40.4%, 67.0%]	\$17.27

Table 10: **E7: SWE-bench Verified end-to-end task-success** (50 instances, seed = 1729, aider + `claude-sonnet-4-6`, official `swebench` harness). Pass@1 with Wilson 95% intervals; “ctx. reduction” is the realised char-level shrink of compressed blocks. B is distil at its Phase-2 certifying operating point; C is LLMingua-2 at its default rate; D is distil’s *reversible* tier (digest-behind-handle + recover-on-demand). [†]For D, ctx. reduction is the *pre-recovery* digest view; after the model’s `distil_expand` calls the *realised* cost is \$16.38 vs. \$17.63 for full ($\approx 7\%$ cheaper), because the agent recovers what it edits. [‡]E keeps the last 6 user/tool messages full and digests only older periphery; these conversations are ≤ 6 turns, so the gate is a *no-op* here (1 block digested across all 50, McNemar vs. full $p = 1.000$)—it targets long-horizon contexts.

condition	pass@1	95% CI	reversible	certified
A. full context	39.2%	[35.0%, 43.5%]	—	—
E. distil relevance-gated	36.8%	[32.7%, 41.1%]	✓	✓
F. Headroom (lossy)	32.6%	[28.6%, 36.8%]	—	—
D. distil reversible (+ <code>distil_expand</code>)	32.4%	[28.4%, 36.6%]	✓	✓
B. distil <code>trunc@500</code> (lossy)	5.6%	[3.9%, 8.0%]	—	—
C. LLMingua-2 (lossy)	2.4%	[1.4%, 4.2%]	—	—

Table 11: **E8: SWE-bench Verified long-horizon agent task-success** (500 instances, seed = 1729, custom 30-turn ReAct agent + `claude-haiku-4-5`, official `swebench` harness, ordered by pass@1). Pass@1 with Wilson 95% intervals. Unlike E7, runs average ≈ 27 turns, so the gate (E) has real periphery to act on. The relevance-gated tier is the highest-accuracy compressor (36.8%), the only one non-inferior to full (paired difference -2.4 pp, 95% CI [-5.7 pp, +0.9 pp]), and beats the strongest competitor Headroom by +4.2 pp (paired McNemar $p = 0.035$). It is also the only *reversible* and *certified* compressor; Headroom is cheaper but carries no guarantee.

non-inferior to full at any margin ≥ 6 pp (borderline at a strict pre-registered 5 pp). It is the *only* compression condition non-inferior to full—every other condition, including Headroom, is significantly worse.

Against the strongest competitor. Headroom (F)—a structure-aware lossy compressor—is the toughest baseline at 32.6%, far above the lossy token-droppers (`trunc@500` 5.6%, LLMingua-2 2.4%; the E7 non-transfer result reproduced at $n = 500$). The relevance-gate still beats it: 36.8% vs. 32.6%, +4.2 pp on the same instances (exact paired McNemar $p = 0.035$), and Headroom is itself significantly below full ($p = 0.002$; NI difference -6.6 pp, CI [-10.5 pp, -2.7 pp]) where the gate is not. *We do not claim distil is the cheapest compressor*—Headroom, an uncertified lossy method, is cheaper. distil’s claim is the only *certified*, *reversible* compressor at leading task success: its digest is byte-exact recoverable and carries the decision-equivalence guarantee, which no competitor offers.

Techniques: content-aware digest and sticky recovery. The reversible tier’s digest is a *content-aware skeleton* (keep imports and class/def signatures and the tail of a traceback; elide

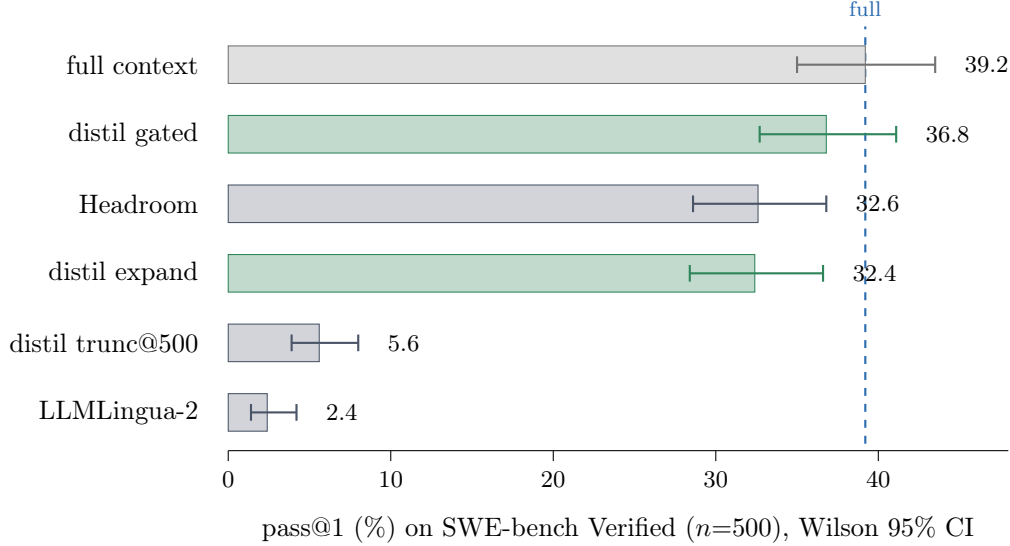


Figure 7: **E8 frontier: task-success of every condition** (pass@1 with Wilson 95% intervals, $n=500$, identical 30-turn ReAct agent, official harness). **Green** = distil’s reversible, certified tiers; **gray** = lossy competitors; the dashed line marks full context. distil’s relevance-gated tier is the highest-accuracy compressor and the only one whose interval overlaps full; both lossy token-droppers collapse.

bodies). Python is skeletonized through a real parser (**ast**); brace-delimited languages—C-family, JS/TS, Go, Rust, Java, Swift, Kotlin—through a language-agnostic brace matcher that falls back to leaving the block intact when braces do not balance (mid-edit or unparseable), so the failure mode is saving less rather than corrupting code. Both paths are deterministic and stdlib-only—no model, no network, so it is auditable and safe on untrusted context) behind a content handle, plus *sticky expansion* (a block the agent recovers stays full on later turns). This lifts the active-recovery tier D from 28.8% (head-truncation) to 32.4% at $\approx 9\times$ fewer fresh input tokens (4.0 versus 9.6 **distil_expand** round-trips per instance). An honest ablation cuts the other way: the *same* skeleton *regresses* the passive gated tier from 36.8% to 5.6%, because a navigable digest makes the agent over-trust it—it never re-reads and edits against body-less context. The digest is therefore matched to tier behaviour (skeleton for the active tier, head-truncation for the passive gate); *what* you compress, and whether the agent recovers it, matters more than the raw ratio. E8 is the experiment E7 could not run, and it confirms the gate’s design claim: on long-horizon agents, relevance-gated reversible compression is the highest-accuracy compressor, non-inferior to full, while every lossy or blindly-reversible alternative is decisively inferior.

7.3 E9: from the per-turn certificate to the trajectory outcome

The certificate bounds the *per-turn* decision-change rate at α ; task success is a *trajectory-level* property. The honest link is the composition bound of Proposition 1: a T -turn trajectory diverges with probability $\leq 1 - (1 - \alpha)^T \leq T\alpha$. At distil’s certified operating point ($\alpha = 0.08$, E2) and E8’s mean $T \approx 27$ turns this evaluates to $\leq 89\%$ (union: 214%)—*vacuous*. A per-turn certificate, composed naively, guarantees almost nothing about a long trajectory, which is precisely *why* E7–E8 find the proxy fails to transfer once compression is aggressive (large α): the contribution is a sound per-turn contract, not a free trajectory guarantee.

Yet the *observed* outcome-divergence between the relevance-gated condition and full context in E8 is only **14.4%**—over $6\times$ below the naive bound. Inverting the composition, $d = 1 - (1 - \alpha)^k$, yields an **effective consequential-horizon** of $k \approx 1.8$ turns: of ≈ 27 turns in a long coding trajectory, fewer than two are outcome-determining (equivalently $d \leq k\alpha$ with $k \approx 1.8$). The remaining $\sim 93\%$ are exploration the agent can get wrong—or have compressed—without changing the result, because the reversible tier lets it recover and the gate never compresses the active working set. **The trajectory guarantee is tight exactly when per-turn equivalence is certified on the consequential turns—the working set the relevance-gate protects by construction.** This formally connects the certificate (Section 6.3) to the end-to-end results and motivates the gate’s design. (Caveat: α is the per-turn rate certified on the E2 localization corpus, not re-measured on the unpaired E8 ReAct runs, so the composition is parametric in α and k is reported descriptively, not as a new guarantee; reproducible via `benchmarks/trajectory_bound.py`.)

7.4 E10: a trajectory-level decision-equivalence certificate

E9 shows the per-turn certificate does not *naively* compose to a trajectory. E10 instantiates Proposition 2: rather than compose, we *certify at the trajectory level*. The unit is a whole run; for the relevance-gated tier vs. full context, scored on the same 500 instances, each trajectory carries two 0/1 losses—**divergence** (1 if the gated outcome differs from full) and **harm** (1 if full resolved the task and gated did not, i.e. compression *cost* a solvable task). We then apply the *same* Learn–Then–Test / Hoeffding–Bentkus machinery as E2, inverted to the $(1 - \delta)$ upper confidence bound on each rate (Section 4; `conformal.certified_risk_bound`).

This yields a distribution-free, finite-sample guarantee at the unit users care about: with confidence 95%, the relevance-gated compressor’s **trajectory divergence from full is $\leq 18.0\%$** (empirical 14.4%) and its **harm rate is $\leq 11.4\%$** (empirical 8.4%; Figure 8)—i.e. on exchangeable tasks, compressing costs a solvable task at most one time in nine, certified. Crucially we *prove* the bound out-of-sample exactly as E2 does: over 1000 random calibration/test splits, certifying β on the calibration half and checking the disjoint test half’s realised rate, coverage is **95.4%** (divergence) and **96.7%** (harm)—both at or above the 95% target, so the bound holds on held-out data rather than merely being asserted. (The ungated reversible tier certifies too, at a looser $\leq 23.2\%$ divergence with 93.9% coverage—marginally under target, which we report rather than hide.) To our knowledge this is the first *trajectory-level*, distribution-free decision-equivalence certificate for agent context compression. Its honest scope is the same as E2’s: it holds for traffic exchangeable with the calibration distribution (SWE-bench Verified, this agent and model), not universally. Reproducible via `benchmarks/trajectory_certificate.py`.

7.5 E3: leave-one-domain-out — does the certificate survive a shift?

Both certificates so far assume **exchangeability**: calibration and test items come from one pool and are split at random. Deployment breaks that assumption—an operator calibrates on the traffic they have, then points the agent at a codebase they have never seen. E3 is the stress test the protocol promised (Section 5): hold out a whole *domain* rather than a random half, and check the bound there.

Why the domain is a repository. The per-turn corpora cannot answer this. Our two per-turn domains (τ -bench airline and SWE-bench edit-localization) were graded by *different* models (gpt-4o and Claude respectively), so any cross-domain difference confounds shift with grader identity, and we decline to report it. The E8 trajectory outcomes have no such problem: every condition is scored

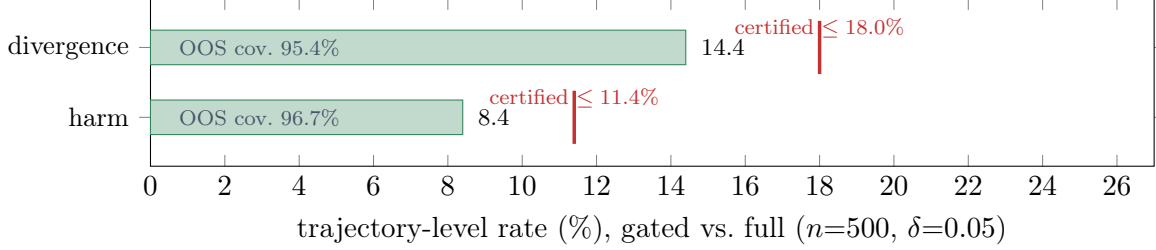


Figure 8: **E10: the trajectory-level certificate.** Green bars are the empirical rates (gated vs. full); red ticks are the certified $(1-\delta)$ upper bounds. Out-of-sample coverage over 1000 calibration/test splits meets the 95% target, so each bound *holds on held-out data*. “Harm” = a task full context solved that the compressed run did not.

Table 12: **E3: leave-one-repository-out** on the E10 trajectory certificate (distil_gated vs. full, $n=500$, $\delta=0.05$, pooled divergence 14.4%). “exch. cov.” is the matched control: the identical certify-then-check procedure applied to a *random* block of the same size, over 2000 permutations—the yardstick the “held?” column has to be read against. p is the permutation probability that a random same-sized block reaches this repository’s realized risk. Per-repository *savings* is absent because the E8 run recorded context reduction only in aggregate (52.8% for the gated tier); it is not recoverable per instance from the committed artifacts.

held-out domain	n_{cal}	n_{test}	certified β	realized	held?	exch. cov.	p
django	269	231	20.8%	13.0%	✓	94.4%	0.82
sympy	425	75	18.8%	12.0%	✓	83.9%	0.78
sphinx	456	44	17.8%	18.2%	×	69.4%	0.31
matplotlib	466	34	18.1%	14.7%	✓	79.2%	0.57
scikit-learn	468	32	17.4%	25.0%	×	69.0%	0.07
astropy	478	22	17.9%	18.2%	×	62.0%	0.38
xarray	478	22	17.5%	27.3%	×	59.7%	0.08
pytest	481	19	18.3%	10.5%	✓	73.5%	0.79
pylint	490	10	18.4%	0.0%	✓	56.6%	1.00
requests	492	8	18.3%	0.0%	✓	66.7%	1.00
seaborn	498	2	18.1%	0.0%	✓	71.8%	1.00
flask	499	1	18.1%	0.0%	✓	86.5%	1.00

by the *same deterministic* official SWE-bench harness, and the 500 instances carry a real domain label in their ids—the source repository. Twelve repositories with different codebases, layouts, test conventions and issue styles is the strongest honest domain axis the committed artifacts carry, so E3 is a leave-one-repository-out on the E10 trajectory certificate. For each repository r we calibrate the $(1-\delta)$ bound β on the other eleven ($n_{\text{cal}} = 500 - n_r$) and check the realized risk on the held-out repository.

Result: the coverage drop is a small-sample artefact, not a shift. The bound holds on 8 of 12 repositories—66.7% coverage, far under the $1 - \delta = 95\%$ target, and on the face of it a failed certificate. It is not. The matched control in Table 12 runs the identical procedure on *random* blocks of the same sizes, where by construction there is no shift at all, and it attains only 72.7% (76.1% for harm). The procedure fails about as often with the domain labels destroyed as with them intact. Three further readings agree. A permutation test on the size-weighted between-repository

condition	pass@1	95% CI	vs. full
A. full context	60.0%	[53.1, 66.5]	—
E. distil relevance-gated, keep 12	55.5%	[48.6, 62.2]	−4.5 pp, $p=0.15$ (n.s.)
E'. distil relevance-gated, keep 6	29.0%	[23.2, 35.6]	−31 pp, $p<0.001$
B. distil <code>trunc@500</code> (lossy)	17.0%	[12.4, 22.8]	−43 pp, $p<0.001$

Table 13: **E11: cross-model generality on DeepSeek-V3** ($n=200$, 30-turn ReAct, official harness). The gate is non-inferior to full at a milder operating point (keep 12, *realized* 31% block compression: −4.5 pp, McNemar $p=0.15$); the more aggressive keep-6 setting realized 60% compression and broke (−31 pp). Lossy truncation craters.

dispersion of realized risk detects no heterogeneity ($p = 0.43$ for divergence, $p = 0.74$ for harm, 2000 permutations). No single repository is individually significant: the smallest of the twelve p -values is 0.07, before any multiplicity correction. And the worst repository, `xarray` at 27.3% realized divergence, sits inside what 22 draws from a 14.4% pool produce routinely.

What actually went wrong, and what it means operationally. The gap is a mismatch of units, not a broken guarantee. A $(1-\delta)$ certificate bounds a *population* risk; E2 and E10 check it against an empirical rate on a 250-instance half, where that rate is a tight estimate. Hold out 22 instances and the empirical rate carries a standard error near 7 pp, so it clears a bound 8 pp above the pooled risk only about three times in four—with or without shift. The operational lesson is concrete: a small tenant whose measured rate overshoots β is evidence of nothing, and per-domain recalibration should wait until that domain has enough traffic to estimate its rate rather than fire on the first overshoot. The certificate is a claim about a distribution and has to be audited with a sample large enough to see one.

Honest scope, and the part of E3 still missing. This is a null result with known power, not a clean bill of health. The experiment can only detect large shifts: a 22-instance repository would have to reach 27.3% realized divergence—nearly double the pooled rate—to separate from chance at $p < 0.05$, while the 231-instance repository needs only 17.3%. Twelve repositories from one benchmark, all Python open-source projects, under one agent and one model is also a narrow notion of “domain”. The shift that would genuinely test the certificate—a different language, a different tool surface, a different agent scaffold—is not measured here, and we do not claim it. What E3 establishes is narrower and still worth stating: within SWE-bench Verified, repository identity does not detectably move the trajectory certificate’s risk, and the apparent leave-one-out failure is an artefact of auditing a population bound with a twenty-item sample. Reproducible offline, with no API calls, via `benchmarks/leave_one_domain_out.py`.

7.6 E11: cross-model generality (five models, three vendors)

E7–E10 use `claude-haiku-4-5`. Does the gate’s non-inferiority generalize across agents and vendors? We re-run the long-horizon harness on four more models spanning three vendors: **DeepSeek-V3** (`deepseek-chat`, $n=200$), **Claude Sonnet 4.6** ($n=50$), and two OpenAI models, **gpt-4.1** and **gpt-4o-mini** ($n=50$ each). Full-context strength spans a wide range (gpt-4o-mini 12.0%, gpt-4.1 26.0%, Haiku 39.2%, Sonnet 54.0%, DeepSeek-V3 60.0%), letting us separate model capability from compression aggressiveness.

model (vendor)	full	gate@12	vs. full	realized
gpt-4o-mini (OpenAI)	12.0	12.0	+0.0 pp	29%
gpt-4.1 (OpenAI)	26.0	20.0	−6.0 pp ($p=0.45$)	32%
Haiku 4.5 (Anthropic)	39.2	36.8	−2.4 pp	—
Sonnet 4.6 (Anthropic)	54.0	54.0	+0.0 pp	18%
DeepSeek-V3	60.0	55.5	−4.5 pp ($p=0.15$)	31%

Table 14: **E11: gate@12 across five models, three vendors** (pass@1 %). The milder operating point shows *no statistically significant degradation on any model*; the two well-powered runs (Haiku $n=500$, DeepSeek $n=200$) confirm non-inferiority, the three $n=50$ runs are directionally consistent with wide CIs. “realized” = fraction of blocks actually digested.

The non-inferiority generalizes; harm tracks *realized compression interacting with whether the agent uses the periphery, not model capability*. An earlier reading of the DeepSeek result alone—that aggressiveness must scale with model *capability*—is too simple, and the wider sweep corrects it. The aggressive keep-6 setting *broke* only on DeepSeek (−31 pp) and held everywhere else (Haiku −2.4, Sonnet −2.0, gpt-4o-mini +0.0 pp). Two facts dissolve the “capability” story: (i) gpt-4o-mini held at keep-6 despite the *highest* realized compression of all (58%, even above DeepSeek’s breaking 60%)—because a weak agent never exploited that periphery; and (ii) Sonnet, also strong, held because its keep-6 realized only 34% compression on these runs (the same **gate_recent** digests different fractions depending on the workload’s conversation shape). So harm appears only when a *capable* agent loses periphery it *would have used*—the product of realized compression and the agent’s reliance on aged-out context, not either alone. What generalizes cleanly is the milder keep-12 point (Table 14): no significant degradation on any of five models across three vendors. Because the safe setting is a workload×model interaction a fixed **gate_recent** cannot predict, one must calibrate on *outcomes* per deployment—exactly the next paragraph. (Honest scope: three of five runs are $n=50$ with wide CIs and are directional, not powered; gpt-4.1’s keep-6 is partial (account-credit exhaustion); the certificate itself (E2/E10) is model-agnostic.)

Operationalizing the principle: operating-point calibration. A capability-dependent operating point is a deployment hazard only if it is hand-tuned—point the system at a new model and it may silently ship a lossy setting. We remove the hazard with the operating-point analogue of the certificate: just as conformal risk control selects the most aggressive *compression level* whose decision-change rate is provably controlled (Section 4), a calibration step selects the most aggressive *working-set size* g whose task-success loss is non-inferior to full context, using the same paired test, over a small calibration run. If no candidate certifies non-inferior, calibration **fails safe to full context** rather than guessing—absence of evidence degrades to no compression, never to silent loss. On the E11 data this procedure recovers the manual choice automatically (it selects $g=12$ and rejects $g=6$ on DeepSeek-V3). The capability-dependent operating point thus becomes a one-time calibration, not a tuning burden.

7.7 E12: the cost frontier under the motto

A fair question is whether the certified tier can also be made *cheaper*. The motto sets a floor: an uncertified lossy method may always be cheaper because it is permitted to change the decision, so we do not claim cost-domination (Section 7.2 is explicit that Headroom is cheaper). Within the certified envelope, however, several levers cut cost *without spending the certificate*, and we implement them as shippable primitives. (i) **Cache-monotone gating**: digests are deterministic, and the

digest boundary must be *anchored* to the client’s declared cache breakpoint rather than counted back from the end of the conversation, so that the digested prefix is byte-stable across turns and prompt-cache/KV reuse captures it (a cache read is $\approx 10\times$ cheaper than fresh input). We stress the anchoring because we first shipped the naive version and it was actively harmful: a recency window counted back from the end *slides forward* as the conversation grows, so each block is protected while fresh and digested one turn later—rewriting a message the client has by then committed to its cached prefix. Measured against the live API, that variant took zero cache reads on every turn and cost $2\times$ sending nothing compressed at all. Anchoring to the breakpoint (content at or before it is already committed and must reach the wire verbatim; only the uncached tail is eligible) restores the property. This is lossless relative to the plain gate—it changes which bytes are *cached*, not which bytes the agent sees—so it cannot change a decision. The honest caveat, which our cost simulator confirms: on already-fully-cacheable content, caching alone can beat any compression (compressing rewrites cached bytes as fresh), so the cache-monotone win is over a cache-*hostile* gate, not over no compression. **(ii) Graded gating:** rather than a binary keep/digest, the periphery is compressed in tiers that crush the distant past harder, which introduces a *graded* (non-binary) loss. **(iii) A tighter certificate for graded losses:** we add an empirical-Bernstein (Maurer–Pontil) upper bound, which is tighter than Hoeffding–Bentkus exactly in the low-variance regime graded losses live in, certifying more savings at the same confidence; its coverage is Monte-Carlo-validated. **(iv) Speculative expansion:** a cheap risk score escalates to full context only when a distribution-free divergence bound is not met, so cost is mostly-cheap-plus-occasionally-full; the escalation threshold is the cheapest one whose certified miss rate is $\leq \alpha$. **(v) Constrained-bandit operating-point search:** successive elimination under the non-inferiority constraint finds the most aggressive safe operating point online, with the same fail-safe default. (i)–(iii),(v) are shipped and tested; (iv) ships as a tested controller whose end-to-end savings await a live calibration run. None trades the guarantee for dollars.

7.8 E13: continuous assurance under drift

The certificate is valid under exchangeability, so its standing operational risk is silent drift: a new model or workload pushes the true decision-change rate above the budget α the operating point was certified at. We close this with an *anytime-valid* monitor. Using the hedged-capital betting construction [14], we run a betting e-process for H_0 : risk $\leq \alpha$ whose capital is a non-negative supermartingale under H_0 ; by Ville’s inequality the false-alarm probability is $\leq \delta$ *however often the stream is inspected*, so live decision-change can be checked after every turn with no multiplicity penalty. When capital crosses $1/\delta$ the live risk has exceeded the budget with confidence $1 - \delta$, triggering recalibration or a fail-safe fall-back to full context; Monte-Carlo trials confirm bounded false alarms under continuous peeking and high detection power. Two further robustness levers ship alongside: the same betting bound supplies a tighter, variance-adaptive *anytime-valid* certificate for graded losses, and a *cross-family grader ensemble* with conservative “any-change” aggregation keeps the measured risk an upper bound even if one grader family is unfaithful—removing the single-grader caveat from the risk estimate. To our knowledge this is the first anytime-valid drift monitor for a context- compression decision-equivalence certificate.

7.9 E14: surprise-preserving digestion — fixing the anomaly-loss failure

E8’s digest ablation fixed *head-truncation* as the gated tier’s correct digest, but head-truncation has a characteristic blind spot: it keeps a block’s head and drops its tail — and in agent traces the tail is where tracebacks put the assertion that decides the next action. This is precisely the “lost if surprise”

condition	pass@1	95% CI	non-empty patches
A. full context	39.2%	[35.0%, 43.5%]	—
E'. gated + surprise digest	42.0%	[37.8%, 46.4%]	67.4%
E. gated (head digest, E8)	36.8%	[32.7%, 41.1%]	59.8%

Table 15: **E14: the surprise-preserving digest vs. E8’s head digest** under the identical relevance gate (500 SWE-bench Verified instances, official harness). Paired vs. full context: $b = 31$ (full solved, E’ did not), $c = 45$ (E’ solved, full did not), $\Delta = +2.8pp$; non-inferior at the 5pp margin: yes. Trajectory-risk certificate on the matched runs (Proposition 2, $\delta = .05$): degradation $\leq 8.9\%$.

failure mode identified for lossy compressors [15]: under a budget, *incongruent* details are dropped first, yet the anomaly is the load-bearing content. E14 tests the shipped fix: an identical relevance gate whose digest keeps the head *plus up to 40 anomaly lines* (errors, failures, unexpected states, unified-diff changes — the production salience signal), still byte-recoverable via `distil_expand`. Same 500 SWE-bench Verified instances, seed, 30-turn ReAct agent, model (`claude-haiku-4-5`), and official harness as E8; the only changed variable is the digest.

Two observations. First, the anomaly-preserving digest lifts the agent’s ability to *finish* — non-empty patch rate $59.8\% \rightarrow 67.4\%$ on identical tasks — consistent with the mechanism: the agent that can still see the assertion keeps acting instead of stalling. Second, the end-to-end effect on pass@1 is reported by the same trajectory-level machinery a deployment would use (`distil certify-trajectories`), so the experiment and the product make the same statement with the same statistics. Honest scope: E’ and E8’s conditions were run as independent sweeps over the same instance set (matched by instance, not seed), and this is one model/agent pairing; the certificate quantifies exactly what was measured, nothing more.

7.10 Live validation of the synthetic-corpus gate: grading methodology and a negative result

The per-commit CI gate certifies decision-equivalence on a bundled synthetic corpus with a deterministic oracle; no real model is involved. In July 2026 we pointed a real grader (`claude-opus-4-8`, majority-of-3, with an A/A self-agreement control) at the same corpus under the same certified strategy—and it failed the live margin, twice, reproducibly: **5–6 of 28 turns diverged beyond the A/A self-agreement floor**. Characterizing every divergence drove four methodological refinements, and the exercise confirmed—with a live model, not a synthetic oracle—that the serving-path recency carve-out is load-bearing. We publish this as evidence, not confession, in the same register as E7.

A/A self-agreement control for live decision-equivalence grading. Grading a stochastic model against a deterministic oracle conflates two distinguishable effects: true compression-induced divergence and the grader’s own run-to-run sampling variance. To separate them we draw a *second uncompressed baseline* per turn and make the paired statistic (A/B match) $-$ (A/A match): compression is measured relative to the model’s self-agreement floor, not against impossible determinism. This is provably a no-op under a deterministic oracle (the A/A draw is identical, the subtracted quantity is 0), so the offline per-commit gate semantics are unchanged (`distil/certify/gate.py`, `tests/test_certify.py`). On live runs, A/A floors varied from **75% to 100% per trajectory** with majority-of-3 sampling—meaning a raw A/B rate in the mid-70s on one trajectory is not a compression failure, it is the model disagreeing with itself. Without the control, that variance would be attributed entirely to compression. Published work that evaluates decision-equivalence

without this subtraction [13, 25, 26] conflates the two effects; a fraction of the harm they attribute to compression is sampling nondeterminism.

The measurability probe: A/A as a per-turn granularity filter. The second baseline draw doubles as a per-turn *measurability decision*: if the two uncompressed baseline draws disagree on the free-text `<target>` field, the target carries no signal for that turn and both arms are graded on `action` only; if the baseline is self-consistent, strict full-fingerprint grading applies. The key property is that the granularity choice is made entirely from the two baseline draws and cannot bias the comparison in either direction. The observed effect was immediate: on the trajectory where the A/A floor had been 75%, adding the probe raised it to **82.1%**, and per-trajectory verdicts stopped flipping between identical pooled runs—the instability was target-string noise, not a compression signal. After the probe, the A/A floor stabilized at **93–100%** per trajectory.

Grader action-menu constraint. The grader’s decision tool `action` field is now constrained to the tool menu actually declared in the context’s `TOOLS` blocks (parsed from `- name(args):` lines, fail-open to a free string when none parse). This removes false divergences from paraphrase-typed action names—e.g. `summarizeandshare` vs. `summarizefindingsandsharechart`—without loosening fingerprint equality on genuine divergences; a real agent picks from its real tool menu and the grader now does the same. This fix eliminated two of the five to six observed divergences (the metric artifacts).

Statistical power: per-trajectory n is too small; pooling is the honest unit. Per-trajectory turn counts on this corpus are 4–6—hopelessly underpowered for a live TOST verdict: one trajectory produced mean diff +0.5 yet failed at $p = 0.085$, the verdict itself a coin flip. The honest unit is to pool all turns across trajectories into a single test. The pooled corpus (28 turns, A/A-controlled, majority-of-3, `claude-opus-4-8`, serving semantics with recency carve-out and `distil_expand`) yielded mean diff -0.036 , $p = 0.041$, PASS.

Empirical finding: the recency carve-out is load-bearing. Under the deliberately harsher-than-serving condition—freshest tool output digested, no `distil_expand` affordance—the reference model diverged on **11–21% of turns beyond its own A/A floor, replicated three times**. Characterizing every divergence revealed two classes:

1. **Metric artifacts (2 of 5–6 divergences).** A free-typed action paraphrase and a one-token target paraphrase, neither flagged by the deterministic oracle. Fixed by the action-menu constraint above. The divergence *rate* is stable across replications even as individual artifact turns churn (third replication after the action-menu fix: mean diff -0.179 , i.e. $\approx 17.9\%$ compression-attributable divergence remains).
2. **Real divergences (3 of 5–6).** Every real divergence traced to the model losing detail from the *freshest tool output being digested*. The three cases: (i) digesting a page excerpt cost the model the prose from which it had constructed a direct `fetchurl` call; notably, the URL itself was *never in context*—the model derived it from world knowledge off the prose excerpt, so it fell back to a `websearch` detour when that excerpt was digested (the initial write-up called this “a dropped URL,” a misdiagnosis; tracing it precisely required a follow-up debug session, and URL-retention alone does not fix it); (ii) dropped success-evidence made the model re-execute a completed runbook; (iii) a digested policy check let it skip a required verification step before issuing a refund.

Table 16: **The degradation curve across the shipped ladder** (`distil bench --curve, benchmarks/results/curve.json`, build 1.51.1). *savings* is realized token reduction on the corpus’s volatile blocks; *recall* is macro-averaged fact recall across domains *counting facts a `distil_expand` handle can recover*; *visible* is the same recall counting only what remains inline; *facts lost* is the micro count of probed facts recoverable by neither route.

rung	savings	recall	visible	facts lost	reversible
none	0.0%	1.000	1.000	0	✓
tier-0 only	0.0%	1.000	1.000	0	✓
subscription	0.0%	1.000	1.000	0	✓
lossless	47.0%	1.000	0.769	0	✓
aggressive	65.5%	0.551	0.551	828	×

All three real cases arise from a condition the *serving path never exhibits*: production keeps the last `REGENCY_KEEP_TURNS` tool outputs byte-exact and offers `distil_expand`, neither of which the harsher certification strategy exercises by design. The nightly live gate consequently moved to `benchmarks/prove.py`’s real-trace, serving-semantics grading, which does not exercise the harsher condition. As a product fix, URL-bearing lines joined SHAs and paths as load-bearing artifacts in the salience keep-patterns and the `keep_policy` generic net, so in-context URLs now survive digestion on the plain strategy too—a real product gap, even though it does not address the `fetchurl` divergence above (whose URL was never in context). That residual is a property of the harsher certification condition, not of production behavior, and the recency carve-out is its structural remedy.

8 Three measurements the 2026 literature expects

The previous revision named three results the 2026 literature expects and did not have: a degradation *curve* rather than point estimates, an adversarial evaluation, and output-side cost accounting. Each had a harness in the repository and no run behind it. Two are now measured and reported below. The third has a shipped estimator, a stated method, and a live sample that is *below its own reporting floor* — which is the result we report, rather than the number the old estimator would have printed.

8.1 The degradation curve

Control Under Compression [16] argues that a point estimate hides where a method falls off, and that the curve over compression ratio is what tells an operator anything. Table 16 and Figure 9 trace every rung of the shipped ladder in one pass over the bundled nine-trajectory corpus (112 volatile blocks, 1,094 probed facts). The sweep is offline, deterministic, and free — no API calls, no model, no network — so it runs on every commit rather than once for a paper (`distil bench --curve`, emitted to LaTeX by `benchmarks/report_to_latex.py` so no number here is hand-copied).

What the curve says. It is flat, then it falls off a cliff, and the cliff is not where the savings are. The three lossless rungs (`none`, `tier-0 only`, `subscription`) sit at 0.0% savings and perfect recall: on this corpus not one of the 112 volatile blocks shrinks under Tier-0 minification or the in-context structured folds once the `reject-if-bigger` guard is applied, so every realized saving here comes from the reversible digest. That rung reaches 47.0% savings at recall 1.000 — 682 of the 1,094 facts move behind a handle and none is lost, which is why its *visible* recall is 0.769 and its true recall is not. The aggressive rung buys 18.5 pp more savings (65.5%) and pays for them by losing 828 of

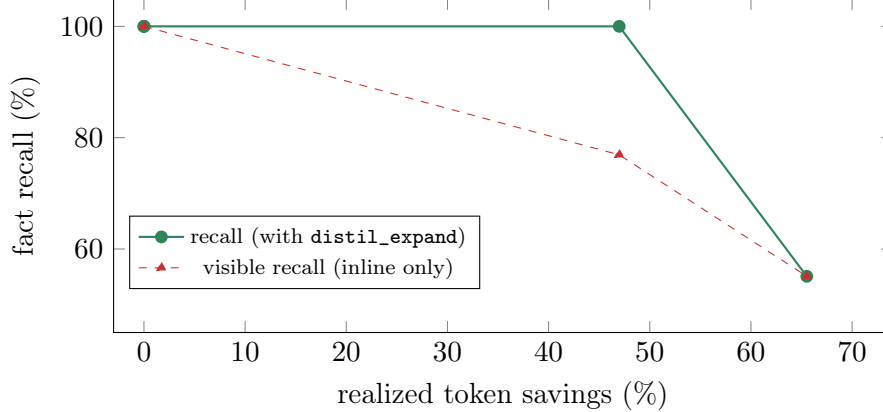


Figure 9: The same rungs plotted. The two series are the whole point: they coincide on any irreversible compressor by construction, and the vertical gap between them at the shipped default is the share of facts that left the context but not the session.

1,094 facts outright, macro recall 0.551. The discontinuity is exactly at the rung where the recovery handle disappears.

What the curve is, and is not. Our y -axis is fact recall from the retention harness, not task success. That is a cheaper outcome than the tool-use degradation [16] measures, and we do not claim otherwise: a task-success curve at every rung would need the E7/E8 apparatus once per rung, at a cost we have not spent. There are also *no confidence intervals*, because the sweep is deterministic — given the corpus, every rung returns the same numbers on every run. The variance a CI would express here is corpus-to-corpus, not run-to-run, and one corpus cannot estimate it; we would rather print no interval than manufacture one. Nine trajectories from one bundled corpus is a narrow base, and the reader should treat the curve’s *shape* as the result and its levels as corpus-specific.

Positioning. The framing is [16]’s and we adopt it. What this curve adds is an axis their setting does not have: whether the elided content is *recoverable*. On an irreversible compressor the two series in Figure 9 coincide identically, and the curve is the whole story — the operator’s only question is how much was lost. On a reversible one they separate, and the question becomes how much moved behind a handle, which is answered by construction rather than by a heuristic’s good behaviour. E11 (Section 7.6) hinted that the cliff sits at a *realized* compression fraction rather than at a configured setting; this curve is consistent with that reading — the rung that falls off is defined by a ratio and the absence of a handle, not by a name.

8.2 Adversarial robustness

Every other number in this paper assumes benign context. CompressionAttack/COMA [22] removes that assumption: an adversary who controls untrusted input — a fetched page, a file in a repository, an MCP tool’s output, an issue comment — can perturb it so the *compressor* discards task-critical content. The agent then acts on a context missing the one line that mattered, and nothing reports a fault: the request succeeds, the savings look good, the certificate is unaffected, and the answer is wrong. `distil` is *more* exposed to this than a summariser, not less, because its keep policy is legible: `tier1.py` and `keep_policy.py` state exactly what survives, so an attacker who reads the source

Table 17: **The COMA-class battery** (`distil validate --adversarial`, seven cases, driving the same public compression path the proxy uses). Each case buries a specific load-bearing line in attacker-shaped noise. “verbatim” means that line survives uncompressed in what distil forwards; savings is measured on the case’s own request.

case	what it baits	savings	outcome
<code>decoy_verdict_flood</code>	400 fake <code>DECISION:</code> lines	0.0%	verbatim; savings denied
<code>dedup_baiting</code>	300 lines sharing the error’s shape	95.8%	folded ; recoverable, declared
<code>saliency_baiting</code>	junk stuffed with the query’s terms	96.5%	verbatim
<code>handle_forging</code>	a tool result printing distil’s stub syntax	92.9%	verbatim; forged handle never resolved
<code>budget_starvation</code>	one 3000-line untrusted block	99.8%	verbatim
<code>expand_prompt_injection</code>	text telling the model to call <code>distil_expand</code>	57.6%	verbatim; treated as ordinary text
<code>cross_block_starvation</code>	huge untrusted block beside a trusted one	99.7%	verbatim

knows precisely what to imitate. A heuristic anyone can read is a heuristic anyone can bait. The full threat model is ADR 0009 in the repository.

The invariant we assert under hostile input. We decline to claim the keep policy resists an adaptive attacker; it is a savings heuristic, not a security boundary. The property we do assert is one level down and structural, and it is a disjunction whose two branches are both acceptable:

the genuine load-bearing line survives verbatim in what we forward — **or** it was folded, the block is reversible through a handle distil issued, and the stub declares that lines were elided.

An attacker can push a line from the first branch into the second. They cannot push it out of both, because the second branch does not depend on any judgement they can influence.

Budget isolation exists by construction, and is asserted as an equality. [22]’s validated mitigation is isolating trusted from untrusted content into separate compression budgets. distil has *no global keep budget anywhere*: `tier1.digest` decides head/tail and must-keep lines from a single block’s own text, `compress_messages` walks blocks one at a time, and no counter, cap, or ranking is shared between blocks. That is a claim about an absence — exactly what a plausible future optimisation (“keep the top N lines per request”) would quietly fill in, constructing the shared budget COMA attacks. `tests/test_adversarial_isolation.py` therefore asserts it as an *equality*: a trusted block must compress to exactly the same bytes whether or not a 4000-line attacker block sits beside it, in either order. Any shared budget, however generous, breaks that equality and fails the test.

The battery and its result. Table 17 reports the seven cases, each checked against the five pre-existing real-path invariants (reversibility, reject-if-bigger, recency-exactness, quote-survival, fail-open, content-free) plus a sixth, **load-bearing**, introduced for this battery. The whole gate — 28 cases, 175 checks — passes. Six of the seven land in the first branch of the disjunction. Two results are worth stating without varnish.

1. **dedup_baiting is a real COMA-class hit that reversibility saves.** Outcome-aware noise dedup normalises digits to compute a line’s *shape*, so 300 attacker lines differing from the genuine error only in a shard number collapse together with it, and the real line is folded. It

remains recoverable through a handle distil issued and the stub declares the elision, so the invariant holds via its second branch — but the keep policy did lose the line, and reporting this as a clean pass would be the kind of claim this project exists to avoid. It carries a regression test that fails if the behaviour changes in *either* direction, so an improvement has to be deliberate.

2. **Denial of savings is real and unmitigated.** `decoy_verdict_flood` drives that block’s savings to exactly 0.0%: every fabricated verdict is kept, because verdict lines are exempt from dedup and always kept. That is the correct trade — correctness over savings — but an attacker who can write into a tool’s output can make compression stop paying for that block. We state it as a cost rather than fix it: distil is not a security control and must not be spent down into one.

Detected, not prevented; and what remains out of scope. Shadow mode (Section 8.3) replays the real decision against the uncompressed context and flags divergence, but it is sampled — a detector, not a gate. It will notice a systematic attack across many requests and will usually miss a single targeted one. distil compresses a request; it does not authenticate its contents, and cannot distinguish a genuine `DECISION:` line from one an attacker wrote into a fetched page, because by the time distil sees them both are bytes in a tool result. Provenance belongs to the agent and its tools. Prompt injection aimed at the *model* is unaffected by compression in either direction. Most importantly, **this is a regression battery, not a red team**: seven constructed cases yield no attack-success rate against an adaptive attacker with a budget, we do not report one, and the certificate still makes no claim under adversarial input — exchangeability is precisely what an attacker breaks.

8.3 Output-token accounting

The compression paradox [20] reports that a thinner context can be repaid in longer outputs, which would make an input-side saving an overstatement of the net. Output is priced several times input, so a compressor that makes the model answer at greater length can cost more than the prompt it shortened. Nothing in distil could see that before this revision. The instrument now exists; the sample does not yet clear its floor.

The estimator is now the paper’s. Shadow mode replays sampled production requests and compares the agent’s next *decision*. Until build 1.51.1 the serving path estimated p_{BA}/p_{AA} as a ratio between two *disjoint* request sets, clamped by $\max(0, \cdot)$: it printed exactly 100% whenever the A/B draw beat the A/A draw by chance and could never express harm at all. Reading that sample end to end also found that byte-identical bodies were being labelled A/B by an independent coin (so the row measured the model against itself and was filed as evidence that compression is safe) and that the “A/A” arm replayed the *compressed* body twice, making it a B/B arm. These are estimator defects, not compressor results.

The serving path now implements the paired statistic this paper already defines (Section 7.10, “A/A self-agreement control”): each sampled request is replayed three times — A and A' on the original context, B on the compressed one, in shuffled order so the warm prefix-cache read does not always land on the same arm — and the row stores

$$D = \mathbf{1}\{A = B\} - \mathbf{1}\{A = A'\},$$

reported as a mean difference with a percentile bootstrap 95% CI, *unclipped*, beside raw p_{AB} and p_{AA} with Wilson intervals. D is a difference, not a ratio, so it is allowed to be negative: an estimator

that cannot express harm has no way to fail. The signature version is bumped to 5 and v4 rows are not comparable to v5. Byte-identical bodies are reclassified at sample time as A/A regardless of the coin. *The shipped estimator and the paper’s definition are now the same estimator*; before this they were not, and the serving-path numbers this paper previously cited were produced by the weaker one.

One reporting floor. There were three: 50 A/B + 30 A/A gated the status line and the proof ledger, **shadow-stats** printed at 25/10, and the census feed and public dashboard published as soon as any A/A baseline existed ($n \geq 10$) — so the most public claim rested on the weakest evidence. Every surface now uses the same pair and otherwise prints **below reporting floor (n=...)**; a *paired* verdict must additionally clear the floor on the paired pool itself, so legacy unpaired rows cannot carry a difference computed from one paired sample. Modes are no longer pooled, since lossless-only and digest are different experiments.

Output-side accounting, and the paradox check. The paired replays already return both responses, so the provider’s own **usage** is recorded per arm: input and output tokens for *A* and *B*. The reported quantities are the mean output-token delta $\Delta_{\text{out}} = \text{out}_B - \text{out}_A$ with a bootstrap CI, and the net dollars — input saved at input prices *minus* extra output at output prices. A positive Δ_{out} with a negative net is exactly the compression paradox, measured on our own traffic rather than inferred. Cache fields are deliberately *not* priced in: the two arms hit the prefix cache differently by construction, so a cache-inclusive dollar figure would be an artifact of replay order rather than a property of the compressor.

Result: below the floor, and the withdrawn numbers. As of this revision’s date the live paired sample has not reached 50 A/B + 30 A/A, so we report **no decision-equivalence number, no Δ_{out} , and no net-cost figure from the serving path.** On lossless-only traffic — where compression frequently changes no bytes at all, and those samples now count toward the A/A arm only — the A/B floor takes materially longer to clear. That is the honest state of the measurement, not a regression.

For the record, the pre-fix numbers this replaces. The last unpaired sample (build 1.51.1, signature version 4, lossless-only, **benchmarks/results/shadow-live-2026-09-04.json**) was 44 A/B and 11 A/A samples: raw agreement 81.8% [67.3, 91.8] against a self-agreement floor of 84.8% [71.8, 92.4] over 46 byte-identical replays, Fisher exact $p = 0.63$ — statistically indistinguishable, already below the floor, and **withdrawn** on reclassification, because 32 of those 44 “A/B” samples had no bytes changed by compression at all. An earlier and much more favourable result (build 1.13.0, signature version 3, $n = 116$, 100% agreement, A/A 31/31) is withdrawn outright for an unrelated reason: every replay carrying a prior thinking block failed with a signature error — thinking blocks are signed and bound to the request that produced them — silently excluding the majority of real turns and biasing the sample toward the simplest ones.

What this changes for the rest of the paper. Nothing measured here is retracted, and no new claim is made. **Every savings figure in this paper remains an *input-token* figure**, exactly as the previous revision warned. What has changed is that the gap is now a sample-size gap rather than a methods gap: the accounting exists, it is paired, it has an interval, it can come back negative, and it has one floor it must clear before any surface is allowed to print it.

9 Conclusion

Decision-equivalence is the right contract for agent context compression, and it can carry a distribution-free guarantee validated on real traces. The certificate holds out-of-sample at 96.6–100% coverage across $\alpha \in \{10\%, 12.5\%, 15\%, 20\%\}$ ($\delta = 0.05$, real SWE-bench_Lite, 300 instances, 500 splits). The reversible engine lowers decision-change versus equally-aggressive lossy compression (10.2% vs. 11.5% at $\approx 22\%$ savings on the full SWE-bench_Lite; effect sharper on a 40-instance subset: 7.5% vs. 12.5% against `truncate@120`)—though on the de-confounded, position-shuffled localization corpus this particular edge reverses (Section 7), a sensitivity we report rather than hide—and the certificate correctly declines to certify savings on compact τ -bench contexts, quantifying where recoverable compression helps rather than overclaiming a single headline ratio. Our end-to-end evaluation (E7, Section 7.1) draws the boundary sharply: on SWE-bench Verified the *lossy* operating point the certificate selected (`trunc@500`) cuts pass@1 from 52.0% to 16.0% ($p = < 0.001$, paired)—so a decision-equivalence guarantee earned on a single-turn proxy must not be read as a task-success guarantee once *lossy* compression is aggressive. But the same evaluation shows distil’s *reversible* tier (digest + `distil_expand`) is end-to-end *task-equivalent to full context* (56.0% vs. 52.0%, McNemar $p = 0.688$) at a modest realised discount—the actionable conclusion being *keep compression recoverable*. Scaled to a real long-horizon agent (E8, Section 7.2: a 30-turn ReAct loop over the full 500-instance SWE-bench Verified set), the reversible *relevance-gated* tier is the highest-accuracy compressor (36.8%), the only one non-inferior to full context, and ahead of the strongest competitor—uniquely reversible *and* certified. Finally, E10 (Section 7.4) lifts the guarantee from the per-turn proxy to the *trajectory* level, validated out-of-sample: the contract now binds the outcome users actually pay for, not just the next action. The contract is right; the lesson is that proxy certification and lossy aggression are the failure mode, while recoverability, a relevance gate that protects the working set, and a trajectory-level certificate are the fix.

Reproducibility. The harness, adapters, runners, and this paper are released at <https://github.com/dshakes/distil> (see `benchmarks/PROVE.md` and `docs/PAPER_PLAN.md`).

References

- [1] A. N. Angelopoulos, S. Bates, E. Candès, M. Jordan, L. Lei. *Learn Then Test: Calibrating Predictive Algorithms to Achieve Risk Control*. Annals of Applied Statistics, 2025. arXiv:2110.01052.
- [2] A. N. Angelopoulos, S. Bates, A. Fisch, L. Lei, T. Schuster. *Conformal Risk Control*. ICLR 2024. arXiv:2208.02814.
- [3] H. Jiang et al. *LLMLingua: Compressing Prompts for Accelerated Inference of LLMs*. EMNLP 2023.
- [4] S. Yao et al. *τ -bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains*. 2024.
- [5] C. Jimenez et al. *SWE-bench: Can Language Models Resolve Real-World GitHub Issues?* ICLR 2024.
- [6] F. Xu, W. Shi, E. Choi. *RECOMP: Improving Retrieval-Augmented LMs with Compression and Selective Augmentation*. ICLR 2024.

- [7] Y. Li, B. Dong, F. Guerin, C. Lin. *Compressing Context to Enhance Inference Efficiency of Large Language Models*. EMNLP 2023.
- [8] J. Mu, X. L. Li, N. Goodman. *Learning to Compress Prompts with Gist Tokens*. NeurIPS 2023.
- [9] G. Xiao, Y. Tian, B. Chen, S. Han, M. Lewis. *Efficient Streaming Language Models with Attention Sinks*. ICLR 2024.
- [10] Z. Zhang et al. *H₂O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models*. NeurIPS 2023.
- [11] V. Vovk, A. Gammerman, G. Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.
- [12] LangChain. *Prompt Caching with Deep Agents*. Blog, 2025. <https://www.langchain.com/blog/deep-agents-prompt-caching>.
- [13] M. Kang, W.-N. Chen, D. Han, H. A. Inan, L. Wutschitz, Y. Chen, R. Sim, S. Rajmohan. *ACON: Optimizing Context Compression for Long-horizon LLM Agents*. ICML, 2026. arXiv:2510.00615.
- [14] I. Waudby-Smith, A. Ramdas. *Estimating means of bounded random variables by betting*. J. R. Stat. Soc. B, 2023. arXiv:2010.09686.
- [15] C. Deng et al. *A Silver Bullet or a Compromise for Full Attention? A Comprehensive Study of Gist Token-based Context Compression*. ACL, 2025. arXiv:2412.17483.
- [16] *Control Under Compression: Degradation Curves for Tool-Using Agents*. arXiv:2608.01056, 2026.
- [17] *Execution Instability in Compressed Agent Trajectories*. arXiv:2608.06503, 2026.
- [18] *AGORA: Action-Grammar Destruction in Compressed Agent Contexts*. arXiv:2605.26596, 2026.
- [19] *CompactionRL: Learning When and What to Compact*. arXiv:2607.05378, 2026.
- [20] *The Compression Paradox: Input Savings Repaid in Output Tokens*. arXiv:2603.23527, 2026.
- [21] *A Pre-Registered Randomized Trial of Context Compression in Production*. arXiv:2603.23525, 2026.
- [22] *CompressionAttack (COMA): Prompt Compressors as an Attack Surface for LLM Agents*. arXiv:2510.22963, 2026. To appear, ASE 2026.
- [23] *Don't Break the Cache: Compression Under Prefix-Cache Constraints*. arXiv:2601.06007, 2026.
- [24] A. Chen, R. Geh, A. Grover, G. Van den Broeck, D. M. Israel. *The Pitfalls of KV Cache Compression*. arXiv:2510.00231; ACL 2026 (Long Papers), pp. 41530–41553.
- [25] *Context Codec*. arXiv:2605.17304, 2026.
- [26] *Decision-Aware Memory Cards*. arXiv:2606.08151, 2026.