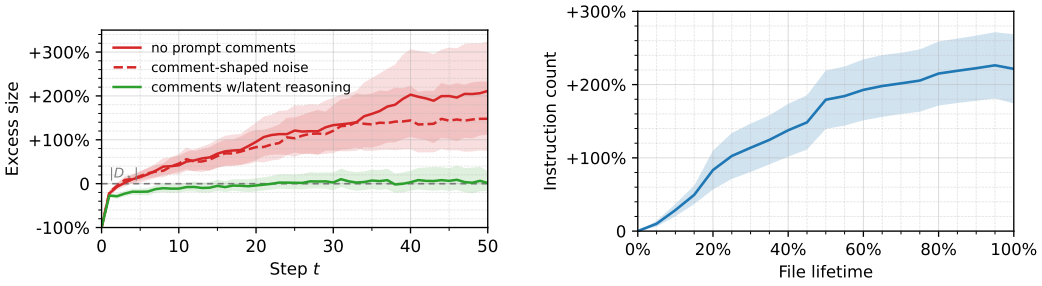
WHY DOES CLAUDE.md KEEP GROWING? CATASTROPHIC REMEMBERING IN AGENTIC CODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic coding READMEs like CLAUDE.md grow without bound in real repositories, stopping only when history ends or someone rewrites the file wholesale. We trace this to imperfect recall: appending an instruction is always cheap, but once an instruction's latent reasoning is gone, deleting it without risking a correctness regression costs $O(2^{|D|})$ in a prompt of $|D|$ instructions. Successive edits destroy that latent reasoning, additions outrun deletions and prompts grow while instruction-following degrades — **catastrophic remembering**. First, we characterize this phenomenon across 247,694 instruction lifetimes in 1,867 repositories: agentic prompts grow nearly monotonically, more than tripling over their lifetime (+226%), gaining +4.7 net instructions every commit, and deletion hazard decays geometrically with instruction age, consistent with imperfect recall and no other tested rival. Then, we show that prompt comments can halt the growth: inverting IFEval yields verifiable worlds whose optimal prompts are known, and there **comments encoding latent reasoning remove 99.3% of excess instructions** (+211.3% to +1.4%). Finally, we show comments **improve instruction-following by up to 23.1%** in real-world prompts and their efficacy grows with model capability. If English is the new code, why don't we have comments yet?



(a) **Causal:** Prompts without comments (or with comment-shaped noise) grow without bound; when comments carry latent reasoning, they settle near the optimal size at equal correctness. Excess size is $\frac{|D_t|}{|D_\star|} - 1$ for optimal IFEval prompt $D_\star$; bands are 95% bootstrap over 184 worlds.

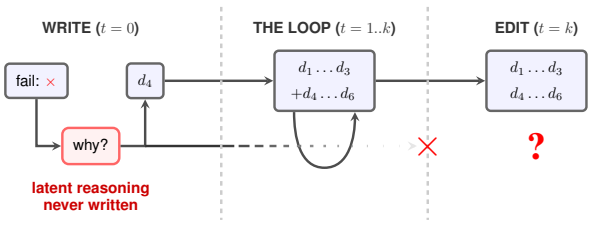
(b) **Observational:** After creation, the number of prompt instructions in an agentic README more than triples (+226%) over the file's lifespan, across 1,867 GitHub repositories and 247,694 instruction lifetimes. Instruction count relative to file's first version, $\frac{|D_t|}{|D_0|} - 1$; lifetime normalized per file.

Figure 1: Deleting prompt instructions requires remembering why they were added; agentic prompts lack comments, so they only grow over their lifetime.

1 INTRODUCTION

Overwhelmingly, agentic READMEs never stop growing. A CLAUDE.md, AGENTS.md, or copilot-instructions.md gains instructions and rarely sheds them (Chatlatanagulchai et al., 2025). Deleting the file is not the answer: agents in repositories carrying one run faster and emit fewer output tokens (Lulla et al., 2026). Keeping everything is not free either: instruction-following degrades as constraints accumulate (Jiang et al., 2024; Lior et al., 2026), and the median file already carries

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Why do agentic prompts only grow? The latent reasoning behind d_i is never written and decays over the (possibly agentic) development loop, so deletion risks an acute regression while the cost of addition stays diffuse.

```
Write a brief post about how
Earth is different from Venus.
...
Write your response as three
sentences, each ending with a
period. # Task 0 continues to
fail on sentence count; last
instruction was vague; maybe
being explicit about periods
will help?
```

(b) The fix: comments carrying latent reasoning encode why an instruction was added, which stops unbounded growth in agentic READMEs.

Figure 2: The prompt-comment world. Latent reasoning is free at write-time and exponentially costly to reconstruct at read-time, but prompt comments preserve it and stop the unbounded growth of agentic READMEs.

39 instructions, the average more than tripling over its own lifetime (+226%, Figure 1b). On real prompts, 16 extraneous instructions cost 24.1pp of constraint satisfaction (Section 4.3).

Different mechanisms can explain this growth. If requirements stop applying (instruction staleness), deletion hazard rises with age. If brittle instructions die young (content fragility), hazard falls. If maintainers lose the rationale (imperfect recall), hazard falls with age and, unlike either rival, geometrically, leaving a fixed share never deleted. Line diffs and file sizes cannot resolve individual instructions over time, so the question stays open.

"What does this code do again?", every engineer has asked. Deleting an agentic instruction risks a regression, and doing so safely requires counterfactuals maintainers cannot run. Recording an instruction's latent reasoning costs $O(1)$, but reconstructing it costs $O(2^{|D|})$ for a prompt with $|D|$ instructions (Section 2). Successive edits destroy that latent reasoning, so deletion hazard decays with age, additions outrun removals, and instructions grow while instruction-following degrades — **catastrophic remembering**. Under catastrophic forgetting, a gradient learner overwrites what it should have kept (McCloskey & Cohen, 1989; French, 1999; Kirkpatrick et al., 2017); under catastrophic remembering, a maintainer keeps what they should have overwritten. Both fail and fail for the same reason: whatever would license the update is gone.

How deletions occur in agentic prompts hints at the cause. 67.2% of deaths arrive in one file-bulldozing commit, after which growth resumes: removing one instruction needs a reason, removing all needs none. The hazard agrees, decaying geometrically with age (Section 3).

Software engineering solved this decades ago with the comment: text addressed to the next maintainer and invisible to the interpreter (Knuth, 1984). Recovering why code was written as it was is developers' most serious problem (LaToza et al., 2006), so the practice is to record the why, not the how or the what (McConnell, 2004). Unsurprisingly, both carry over to agentic prompts.

We show that prompt comments encoding an instruction's latent reasoning can halt unbounded prompt growth, removing 99.3% of the excess size (+211.3% to +1.4%) over 51 steps (Figure 1a). Because irrelevant context distracts models (Shi et al., 2023), this improves instruction-following on real prompts by as much as 23.1% (11.6pp). Ablating comment payloads shows outcome-grounded latent reasoning drives the gain. Finally, the advantage only grows as (agentic) maintainers grow more capable (Section 4).

We contribute a new approach to prompt maintenance, characterized across four dimensions:

- *Theory of prompt maintenance.* Decaying recoverability of latent reasoning predicts unbounded growth (Section 2).
- *Root cause of prompt growth.* Deletion hazard decays geometrically with age over 247,694 lifetimes, consistent with imperfect recall and no other tested rival (Section 3.4).
- *New evaluation with known optimum.* Inverting IFEval yields verifiable worlds whose minimum cover is known, so excess size and correctness become measurable (Section 4.1).



Figure 3: Growth, wholesale rewrite, regrowth — agentic prompts almost never shrink incrementally. Instruction count D_t for six illustrative context files; red rules mark commits cutting at least 75% of a file holding at least 5 instructions. Such rewrites carry 67.2% of instruction deaths; the median file is cut by 73.1% and sits 60.7% below it after 10 commits (52 files, Figure 4).

- *Practical solution.* Prompt comments remove 99.3% of the excess size (Section 4.2), and improve instruction-following on real prompts by as much as 23.1% (Section 4.3).

2 Forgetting Requires Remembering Why

We model prompt maintenance as the *online estimation of an unobservable constraint set from censored, noisy feedback*. Deleting an instruction requires the reason it was written, and that reason decays with age; if it decays geometrically, a fixed fraction of every instruction ever written is never deleted and the prompt grows linearly without bound. Section 3 tests the geometric form.

A task $j = (o_j, C_j)$ pairs a stated objective o_j with its unobservable constraints $C_j \subseteq C$, drawn from a fixed hidden set $C = \{g_c\}$ of verifiers. The prompt at time t is a finite set of instructions D_t; instruction d enters at time τ_d and has age $a_d = t - \tau_d$. A response to j satisfies $c \in C_j$ with probability $q_c(D, j)$, and C is observed only through those draws. At each timestep a (possibly agentic) *maintainer* updates D_t, and the *harness* draws tasks, runs them through the *executor* under D_t, and verifies them against C_j.

The maintainer's target is the **minimum cover** $D_\star$, the smallest instruction set that maximizes expected constraint satisfaction:

$$D_\star = \operatorname*{argmin}_{D \in \mathcal{M}} |D|, \qquad \mathcal{M} = \operatorname*{argmax}_{D} \mathrm{E}_{j,\, c \in C_j}[q_c(D, j)] \tag{1}$$

Every experiment in this paper scores D_t against $D_\star$ on two axes: *correctness*, the expected satisfaction $\mathrm{E}[q_c]$, and *excess size*, $\frac{|D_t|}{|D_\star|} - 1$. A maintenance regime wins only by improving one axis without degrading the other.

We assume only that q_C meets the following properties:

	Property	Behavior
A1	instructability	some d raises q_c
A2	interference	adding d lowers other $q_{c'}$
A3	redundancy	a second d' adds no q_c
A4	censoring	outcomes don't name their d
A5	stochasticity	$q_c < 1$ even when covered

Properties A1–A3 (instructability, interference, and redundancy) set the overall regime whereas properties A4 (censoring) and A5 (stochasticity) respectively block deletions and drive additions.

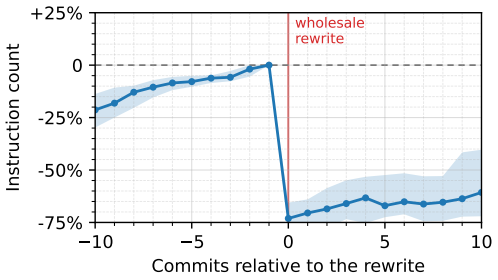


Figure 4: The ratchet: a rewrite resets a prompt's size but not its growth rate. Prompts grow 2.8% per commit before their first rewrite, are cut by 73.1%, then regrow immediately and faster, at 4.8% per commit. Median instructions relative to the *first* pre-rewrite, each file's rewrite event aligned to $t = 0$; band: 95% bootstrap CI.

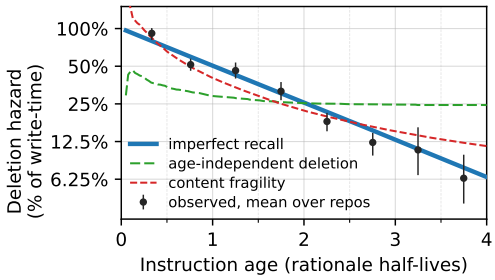


Figure 5: Deletion hazard halves repeatedly — the imperfect-recall signature, which age-independent deletion (flat) misses everywhere and content fragility (bending) past two. Age and hazard are rescaled by each repository's own decay (Table 6). 284 repositories observed to four half-lives, 15,574 deletions, 95% bootstrap.

Deleting without risking a regression requires an infeasible number of counterfactuals. Write d's contribution to c as $\Delta_c(d \mid D) = \mathbb{E}_j[q_c(D, j) - q_c(D \setminus \{d\}, j)]$ over tasks with $c \in C_j$; d is *excess* when $\Delta_c \leq 0$ for every c. Estimating it means probing $D \setminus \{d\}$, and redundancy defeats one-at-a-time probes: two instructions covering one constraint each look free alone, though deleting both breaks it. An honest audit costs $O(2^{|D_t|})$ subset probes, and $D_\star$ is incomputable in general. Knowing why the maintainer added d — its *latent reasoning* r_d — collapses that to $O(1)$, but r_d decays with a_d.

Recoverability sets the deletion hazard. Define the decay

$$\rho(d, a) = \Pr(r_d \text{ recoverable at age } a), \tag{2}$$

with marginal $\bar{\rho}(a) = \mathbb{E}_d\, \rho(d, a)$, so that $\bar{\rho}(0) = 1$ and $\bar{\rho}(a) \to 0$ as $a \to \infty$. We call that decay **imperfect recall**: the instruction stays and its failures recur, but its latent reasoning is increasingly unrecoverable. A maintainer rationally deletes only when the reason survives and the instruction verifies as excess, so the hazard in its own age factorizes (the event-history form used for written organizational rules; Schulz, 1998; March et al., 2000; Zhou, 1993),

$$h(a) \;\approx\; \bar{\rho}(a)\, s(a), \tag{3}$$

for $s(a) = \Pr(\text{excess} \mid a)$. We take the forgetting curve to be geometric, $\bar{\rho}(a) = e^{-ba}$, for forgetting rate b and a rationale half-life of $\ln 2/b$ commits, and leave the excess process free: everything below needs only $s(a) \leq 1$.

An instruction survives to age a with probability $S(a) = \prod_{k \leq a}(1 - h(k))$. With A additions per commit and deletions independent across instructions,

$$\mathbb{E}\,|D_t| \;=\; A \sum_{a<t} S(a). \tag{4}$$

The prompt settles iff the expected lifetime $\mathbb{E}[L] = \sum_a S(a)$ is finite, at $|D_\infty| = A\,\mathbb{E}[L]$, which requires $\sum_a h(a) = \infty$. Under geometric decay it does not, whatever the excess process: $\sum_a h(a) \leq \sum_a e^{-ba} = 1/(1 - e^{-b}) < \infty$, so $S_\infty = \prod_k (1 - h(k)) > 0$ and

$$\mathbb{E}\,|D_t| \;=\; A\, S_\infty\, t + O(1). \tag{5}$$

Equation 5 is catastrophic remembering in closed form: as latent reasoning is lost, a fraction S_∞ of every instruction ever written is never deleted, deletions per commit saturate at $A(1 - S_\infty)$ while additions continue at A, and excess size grows linearly with no change in task or constraints. Recoverability is thus necessary and, as we show later, also sufficient for a bounded prompt.

3 Prompts Ratchet: Growth, Rewrite, Regrowth

Agentic context files grow until someone rewrites them wholesale and then they grow again, characterizing the unique sawtooth-like growth curve we call the **ratchet** (Figure 4). Its deletion hazard and covariates identify its root cause for the first time — imperfect recall.

We decompose agentic prompts into individual instructions and track their decay over commit histories (Section 3.1-3.3), then test instruction staleness, content fragility and effort dilution against the geometric hazard derived in Section 2 (Section 3.4). Our corpus spans 1,867 GitHub repositories, 1,801 multi-version files, 299,440 version-to-version transitions, and 247,694 instruction lifetimes; Appendix A details construction, segmentation, matching and censoring, and Appendix A.3 the sensitivity of every result below to the rewrite rule that censors them.

3.1 Agentic prompts grow unbounded until a wholesale rewrite

Maintainers add and almost never remove, so agentic prompts grow without bound in instruction count, instruction complexity and total size until history ends or a file is rewritten wholesale.

At its last tracked version, the median file holds 39 instructions (and 131 at 90%-ile), well past the threshold at which instruction-following degrades (Jiang et al., 2024; Lior et al., 2026). Among 1,576 multi-version repositories, 64.3% grow their instruction count against 26.6% that shrink it, a median net gain of +7 instructions. On average, excluding mass rewrites, each commit adds a net +4.7 instructions across 19,267 commits.

Every component we measure grows. Over its lifetime, a file gains +226% instruction count (Figure 1b) and +10% in instruction length on average. Alongside, in the repo, the total size of agentic prompts grows +140%, so the count is not text migrating between files (Appendix A.2). Instruction count is a proxy for the executor's instruction-following cost; Section 4.3 measures it directly.

3.2 Most disappearances are not deletions

When maintainers delete instructions, they generally do so wholesale in a mass rewrite. 67.8% of instruction deaths arrive in a rewrite or a migration, where a line diff cannot separate deletion from rewording. This is partly why prompt-change studies report additions dominating without settling any one instruction's fate (Tafreshipour et al., 2025).

A *rewrite* is a commit killing at least 75% of a file holding at least 5 instructions; a *migration* is text reappearing in a sibling file. We censor both as competing risks (84,233 and 682 deaths; Appendix A.3), since maintainers don't judge individual instructions in either case (March et al., 2000). Matching validates at 1.000 precision and 0.933 recall on 50 hand-annotated transitions, leaving 247,694 lifetimes and 40,355 tracked deletions.

3.3 Regrowth resumes faster after a rewrite

Although a mass rewrite resets a prompt's size, it does not address the underlying root cause, so it simply starts growing again afterwards.

Aligning 52 files on their first rewrite at $t = 0$, the median instruction count rises monotonically through $t = -1$, is cut by 73.1% at $t = 0$, and by $t = 10$ regrows to 60.7% below its pre-rewrite size (Figure 4). Agentic prompts grow faster *after* a rewrite: +4.8% instructions/commit (after) vs. +2.8% instructions/commit (before).

3.4 Deletion hazard falsifies rival mechanisms

The deletion hazard $h(a)$ and its covariates identify imperfect recall as the mechanism driving unbounded prompt growth.

First, we show that the decline is geometric. Hazard falls with age, within files as well as across them (slope -0.043 ± 0.011 per commit; Appendix A.2), and on each repository's own decay clock it halves once per rationale half-life (Figure 5). Fitting Equation 3 at constant excess two-stage — within each repository, then across them by random effects — gives $s = 0.072$ and $b = 0.222$ (0.222 ± 0.024) at $R^2_w = 0.815$; this then implies a rationale half-life of 3.1 commits and, by Equation 5, 75% of instructions never deleted. Other common statistical families produce a weaker fit: constant $R^2_w = -0.313$, Weibull 0.291, Burr -0.739 (Table 6).

Next, we show that typical mechanistic rivals can't explain observed phenomena. Instruction staleness, the outdated requirements and comments of Lehman (1980) and Panthaplackel et al. (2021), predicts

Table 1: Comments carrying latent reasoning cut excess size at parity satisfaction. Excess is $|D_T|/|D_\star|-1$ at $|D_\star|=2.2$; rate is the share of a round's scored constraints passed, $t{=}T$ the last-3-round mean. **Bold** = best arm, CIs disjoint; $\pm$ is half a 95% bootstrap CI, asymmetric.

				Instruction Count		Constraint Satisfaction	
Role	Arm	T	N	$\lvert D_T\rvert$	Excess Size (%)	Rate (%, $t{=}T$)	Rate (%, $t{=}0$)
CONTROL	no prompt comments	51	184	6.6	$+211.3 \pm 105.3$	44.0 ± 4.9	23.9 ± 3.9
PLACEBO	comment-shaped noise	51	184	5.3	$+147.9 \pm 79.9$	42.6 ± 4.9	22.3 ± 4.2
TREATMENT	comments w/latent reasoning	51	184	2.2	$\mathbf{+1.4 \pm 22.1}$	44.0 ± 4.7	21.7 ± 3.9

hazard rising with age. Effort dilution — a fixed deletion effort spread over a growing file — predicts deletions per commit scaling with file size at exponent 0, against 1 for per-instruction risk; we estimate 0.74 (0.74 ± 0.13; Appendix A.2), so fixed effort fails but partial dilution may contribute. Content fragility — fragile instructions dying young — bends hazard downward by composition, but its own form, a gamma mixture over a constant hazard (Vaupel et al., 1979), fits at $R^2_w = -0.482$; a frailty shared within each repository's files absorbs 4.8% of the slope (Appendix A.2).

Although indicative, this evidence is correlational; we next assign latent reasoning directly.

4 Comments Halt the Ratchet and Buy Back Instruction-Following

Prompt comments can permanently halt the unbounded growth of instructions in agentic prompts and, in doing so, improve instruction-following through smaller, more robust prompts.

To evaluate prompt effectiveness, we first propose a new evaluation for agentic prompts in task-stationary settings (Section 4.1). We use this evaluation to demonstrate that prompt comments enable prompts to settle near their theoretically optimal minimum cover (Section 4.2). Finally, we show that prompt comments can buy back instruction-following lost to extraneous, noisy instructions in real-world prompts (Section 4.3).

4.1 Inverting IFEval makes the minimum cover observable

In practice, maintainers in agentic coding contexts must learn an effective prompt to satisfy a set of unobservable constraints. In order to maximize correctness, this prompt must simultaneously cover all constraints while, given well-known issues with instruction-following decay (Jiang et al., 2024; Lior et al., 2026), being minimal.

Evaluating a prompt therefore requires measuring both its excess size and its correctness, but calculating excess size requires minimum cover, which is incomputable in general (Section 2). Standard instruction-following suites offer a path.

By inverting IFEval (Zhou et al., 2023), we can attack the problem. For each benchmark item (D_j, C_j) — instructions D_j, verifiers C_j — we apply the following transform:

1. *Hide* D_j. The item's stated instructions become the reference cover $D_\star$, which we take as minimal by construction and do not verify, so excess size is measurable.
2. *Keep* C_j. Its verifiers become the world's hidden constraint set, run by the harness alone and never named to the maintainer.
3. *Generate brief* o_j. A stronger model lossily summarizes D_j as a general task objective ("draft a product announcement").

A fresh maintainer then iteratively reconstructs D_j from o_j at each of T steps, seeing only censored, noisy feedback from C_j. Its arm decides whether it may annotate an instruction d with a comment r_d carrying that instruction's latent reasoning. The next maintainer reads D_t and every r_d; the executor reads only D_t. Implementation details are in Appendix B.

```
Do not truncate or cut off your response; always complete every sentence and thought
you begin.
# r1: response was truncated mid-sentence ("A body that you often s") suggesting
response generation stopped prematurely; may indicate token limit, instruction
conflict, or assistant aborting output; this directive ensures responses are complete
```

```
Use the word 'wilderness' exactly 5 times in your response.
# r3: task 0 failed with 'certain wording is not used the right number of times' — the
response uses 'wilderness' only 2 times but requirement expects exactly 5 occurrences;
d12 (3 bold sections) was passing, so keeping that approach
```

```
Divide the response body into exactly 4 paragraphs: (1) introduction to the water
cycle, (2) evaporation and condensation, (3) precipitation and collection, (4)
conclusion. Separate each paragraph with exactly three blank lines.
# r3: "response is not split into the right number of paragraphs" recurring across
r2-r3; previous directive d18 specified two blank lines between paragraphs but failed;
increasing to three blank lines while maintaining strict 4-paragraph structure
```

Figure 6: Each comment names the failure behind its instruction, a hypothesis, and its outcome. Comments are green (#) and never reach the executor. 3 of 8,541 additions, one per constraint family.

4.2 Comments encoding latent reasoning settle the prompt at its cover

Prompt comments encoding an instruction's latent reasoning settle the prompt at its minimum cover (Figure 1a). Its latent reasoning summarizes the failure behind its instruction, a hypothesis, and how it has fared (Figure 6).

Across 184 histories at $T{=}51$ (one seed), commented prompts end at +1.4% excess size against +211.3% uncommented at satisfaction parity (Table 1), removing 99.3% of the ratchet. A shorter-horizon design replicates it at 3 seeds (Appendix B.2).

Ablations and failed protocols locate the effect in three places (Appendix B.3). The comment's content carries it, not its channel: comment-shaped noise of the same form and length lands inside control's interval (Table 1). To be effective, the content must record the failure it's trying to correct (dropping recurrence count forfeits 37%; Table 11), its latent reasoning and hypothesis (dropping it forfeits 30%) and the evidence to support that hypothesis (narrating attempts without outcomes ends at +70.0%, above its own control; Table 12). We also report all five protocols that failed before COMMIT (Table 12): bounded notes slow growth but plateau above cover, a harness-written digest reaches cover only by emptying prompts, and hedged comments suppress additions indiscriminately.

Stronger maintainers ratchet harder and gain more: across 3 maintainer tiers of one model family, executor fixed, uncommented excess rises from +67.7% to +571.9%, and at the top tier comments improve both size and satisfaction (Table 2).

The protocol over-prunes, emptying 12.1% of prompts against control's 2.4% at a satisfaction cost there (Appendix B.4), and raises maintainer input tokens (7.0M to 9.9M across arms). Covers hold two or three instructions against a median file's 39; Section 4.3 tests scale.

4.3 In real prompts, comments buy back instruction-following lost to noise

Extraneous, noisy instructions degrade compliance with the true, correct instructions, and comments recover most of that loss.

Inverting WildIFEval (Lior et al., 2026) the same way carries the test to real prompts, and to instruction-following rather than count. We convert every benchmark item (D_j, C_j) to a world as in Section 4.1, but seed the maintainer with $|D_j|-K$ true instructions and G noisy ones drawn uniformly from other items' sets $D_{i\neq j}$ rather than with all $|D_j|$. WildIFEval's human-written constraints ship with no code verifiers, so an arm-blind LLM judge scores them, one constraint and response at a time. Across 64 worlds, for $K{=}1$ and $G{=}16$, those noisy instructions cost 24.1pp of correctness on the true instructions already in the prompt, 65.6% at $G{=}0$ against 41.5% here (-24.1 ± 9.2pp; Appendix C.1).

Table 2: Stronger maintainers ratchet harder but comments help them more. Only the maintainer varies (executor: Haiku 4.5). Δ: COMMIT against its own row's control; $\pm$: half the 95% bootstrap CI over worlds; **bold**: largest Δ whose CI excludes zero.

			Uncommented control		COMMIT vs. control	
Maintainer	T	N	$\lvert D_T\rvert/\lvert D_\star\rvert$	Sat. (%, t=T)	Δ Excess Size (%)	Δ Sat. (%)
Haiku 4.5	15	64	1.68 ± 0.58	40.9 ± 7.7	-42.1 ± 21.2	-4.5 ± 12.2
Sonnet 5	15	64	3.07 ± 0.47	38.5 ± 7.6	$\mathbf{-61.7 \pm 6.0}$	$+7.4 \pm 17.1$
Opus 5	15	64	6.72 ± 0.88	45.3 ± 8.7	-52.3 ± 5.9	$\mathbf{+18.4 \pm 19.2}$

Comments lift satisfaction from 50.4% to 62.0% over 3 maintenance rounds (11.6pp, 11.6 ± 6.6pp), a 23.1% relative gain, against an uncommented maintainer on the identical prompt. The latent reasoning in the comment drives this again: comment-shaped noise lands 2.7pp from the uncommented arm (2.7 ± 7.3pp, covering zero). A second-vendor judge re-scoring the same 6,336 verdicts confirms every contrast, nulls included, weakening only the recovery contrast, which keeps its sign without its interval; the two judges' effect sizes differ by 3.8pp (7.8pp against 11.6pp; $+3.8 \pm 5.8$pp, covering zero; Appendix C.3).

Under assignment, then, prompt comments encoding latent reasoning reduce instruction count to its reference cover and buy back instruction-following in real-world prompts. Next, we contextualize and contrast our approach with previous work.

5 Related Work

Empirical studies of agentic context files. Prior measurement establishes that context files earn their keep and only grow. Chatlatanagulchai et al. (2025) characterize `AGENTS.md`-class files, which accumulate content and rarely shed it; Tafreshipour et al. (2025) track prompt evolution in repositories, where additions are the most frequent edit type; and Lulla et al. (2026) associate them with faster agent runs and fewer output tokens. We measure that growth over the lifetimes of individual instructions, fine enough to estimate a deletion hazard and identify what drives it (Section 3.4).

Agent memory and eviction. Agent memory systems already implement forgetting, each against an observable staleness proxy. MemGPT evicts on context overflow (Packer et al., 2023); Mem0 deletes on detected contradiction, and its graph variant and Zep timestamp superseded entries invalid (Chhikara et al., 2025; Rasmussen et al., 2025); FSFM catalogs mechanisms from passive decay to safety-triggered deletion (Gu et al., 2026). Authored instructions admit no such proxy: an instruction does not become outdated merely by being old or rarely triggered. Instead, we show that an instruction's latent reasoning should be the primary driver of whether it should be retained or not (Section 2, Section 3.4).

Organizational behavior. Organizational written rules are artifacts people maintain for decades and have been studied extensively. Lehman's laws attribute a program's growth to a changing environment (Lehman, 1980), the demand-side account; Zhou (1993) treats rule change as a stochastic process whose rate depends on a rule's own age; Schulz (1998) shows rule birth rates falling as the population densifies; and March et al. (2000) assembles both into an account of rule births, revisions, and suspensions. We use their insights to characterize the unique dynamics in agentic programming beyond compositional effects and staleness and further decompose a novel latent reasoning recoverability factor $\bar{\rho}(a)$ that depends on age a (Equation 3, Section 3.4).

Software engineering best practices. Software engineering solved comparable challenges decades ago via syntactic structures and engineering conventions. Literate programming argued that a program is addressed to a human reader (Knuth, 1984); McConnell (2004) makes commenting the *why* standard practice; LaToza et al. (2006) find recovering that rationale is developers' most serious problem; architecture decision records institutionalize recording it at decision time (Nygard, 2011); and commit messages exist to carry it and routinely fail (Tian et al., 2022). We directly build on both the structural

and semantic insights to drive the core pillars of our result, showing that comments encoding an instruction's latent reasoning cut excess size at parity constraint satisfaction (Section 4.2).

However, code-comment fidelity is notoriously weak and entire lines of research are dedicated to their divergence. Comments and the code they describe co-evolve poorly (Fluri et al., 2007); detecting and addressing comment–code inconsistency are their own ML-for-SWE tasks (Panthaplackel et al., 2020; 2021). Our ablations identify that low-information, poorly-structured and misleading rationale carries the same deleterious effects into agentic coding (Section 4.2, Table 11).

Continual learning. Continual learning is our closest cousin but primarily targets tasks with drifting objectives. Networks trained on a task sequence overwrite what earlier tasks taught them (McCloskey & Cohen, 1989; French, 1999); rehearsal and regularization toward parameters that mattered before (Kirkpatrick et al., 2017) preserve what a shifting objective would destroy; L2P, DualPrompt, and CODA-Prompt move that burden into the input, learning *fixed-size* prompt pools a frozen backbone selects from, which bounds growth by construction (Wang et al., 2022b;a; Smith et al., 2023). Although our task is stationary, we borrow their frameworks and find complementary insights driven by fixed resource constraints: fixed parameters driving catastrophic forgetting under shifting objectives there, intermittent failures driving catastrophic remembering under (approximately) fixed instruction counts here (Equation 5, Section 4.2).

Instruction following. IFEval scores responses against verifiable constraints (Zhou et al., 2023); FollowBench grades progressively added constraint levels and WildIFEval collects real requests carrying many at once, both documenting that following degrades as the required set grows (Jiang et al., 2024; Lior et al., 2026). Those benchmarks price the constraints a response must satisfy, the harm our motivation rests on. We invert them to price extraneous instructions instead (Section 4.3) and to make the minimum cover observable, without which excess size is unmeasurable (Section 4.1).

6 Discussion

If English is the new code, why don't we have comments yet?

We show two things: (i) how agentic prompts grow is consistent with an instruction's latent reasoning decaying faster than the instruction, and with no rival we test (Section 3.4), and (ii) under censored feedback and a maintainer-only comment channel, saving that latent reasoning at write time reduces instruction count and wins back instruction-following (Section 4.2, Section 4.3).

Neither half should surprise us. Software engineering identified and solved the same problem decades ago and continual learning is working on solving its dual — catastrophic forgetting — right now. Although both settings differ in detail, both offer lessons in principle; however, LLM agents started driving real work before we could adopt their lessons. We eagerly borrow from both now.

Our results open three novel lines of work. Coding agent developers can give prompts a comment syntax, so an instruction's latent rationale can reach the next maintainer; our prototype, while promising, is just a prototype. Maintainers who manage agentic coding prompts can identify best practices: while they may overlap substantially with software engineering best practices, there are almost certainly differences. Finally, researchers can look for what (i) beats it, since prompt maintenance is continual learning over text and nobody has built its analogue of rehearsal or regularization, and (ii) measures it, at representative constraint counts and for constraints that are not mechanically verifiable.

Every field is borrowing from AI right now. We should borrow back just as readily. The answer to catastrophic remembering was forty years old and one field over, and the next one might be too.

AI Use Statement

We use LLMs liberally as a delegated "compiler": we supply the theoretical model, experimental framework, falsifiable hypotheses, final judgments, coding and experimental standards, etc. and ask LLMs to carry out the execution under the constraints we specify.

Specifically, we used generative AI tools to formulate mathematical claims (e.g. formalizing derivation of Section 2 whose dynamics we outlined in prose), provide feedback on experiment design

(e.g. reviewing our experiment designs), and implement methods (e.g. the experiment and analysis code). To identify the optimal prompt comment protocol, we used an auto-research pipeline that autonomously and iteratively analyzed data, interpreted results and proposed and refined hypotheses given a seed set of initial hypotheses. Our methods also use LLMs as declared in-method tools: one summarizes each Inverse-IFEval world's instructions as its task objective (Section 4.1), and an arm-blind LLM judge scores constraint satisfaction on WildIFEval (Appendix C.1).

We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks or clean and reformat dataset. Three tasks are not applicable to this work: provide critical ingredients for proving mathematical claims, assist in the writing of proofs, and assist with translation.

Additionally, we used generative AI tools to create or modify scientific figures or images (e.g. the plotting scripts behind each figure), create or edit software code (e.g. the scripts that build the paper and generate its numbers), draft parts of a research paper (e.g. first drafts of sections and appendix text, which we then rewrote), summarize or analyse existing literature (e.g. for Section 5), edit a research paper to improve readability (e.g. tightening prose), identify relevant literature (e.g. surfacing candidate related work), format references (e.g. BibTeX entries), and propose a title or keywords for a research paper (e.g. candidate titles).

We have validated all AI-assisted work. We hand-checked all mathematical claims, rewrote virtually all main-body text, and code-reviewed all experiment and analysis code. A generator script emits every table, figure and reported number from the canonical experimental artifacts. Outside the declared in-method uses above, no generative tool generates or manipulates data. Finally, we screened the manuscript with Pangram for inadvertent plagiarism.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics Statement

We mine public repository histories, reduce commit authorship to a count of distinct editors, and release no identities or repository content. Our recommendation carries the main risk: automated rationale-based deletion will remove instructions whose rationale was real, and our protocol empties 12.1% of prompts (Appendix B.4). Deployments should keep a person in the deletion path and exclude safety-relevant instructions.

Reproducibility Statement

Table 3 pins every package a reported number passes through to the canonical run's `env.lock`. The lower block lists our own estimators (Appendix A.2) and the 20 reimplemented verifiers, which return censored complaints against a maintained prompt.

We train no model. The corpus pipeline runs on CPU containers (8 vCPU, 16 GiB) over 1,867 repositories. The controlled experiment's canonical run cost 49,680 model calls, 37.9M tokens and 45 model-hours across roles, seeds and arms. Call counts match across arms by construction; only maintainer input tokens differ (7.0M to 9.9M). We pin model versions, decode at API defaults, and ran no hyperparameter search.

The sampling frame is the repository list of Chatlatanagulchai et al. (2025), which carries no license, so we use only its URLs and re-derive all content from public histories. IFEval (Zhou et al., 2023) is Apache 2.0. We redistribute no repository content and release only derived per-instruction tables and code.

The corpus is 1,867 repositories, 1,801 multi-version files, 299,440 tracked transitions and 247,694 spells, 40,355 ending in deletion (Table 4). We did not measure language, domain, or author demographics.

Table 3: The packages each reported number passes through, at the versions frozen in the canonical runs' `env.lock`, identical across runs. Lower block: our own implementations (Section A.2, Table 7). Satisfaction rates are against our verifiers, not published IFEval numbers.

Package	Version	Determines	Where
`rapidfuzz`	3.14.5	cross-version match at similarity ≥ 70: every spell, death and hazard number	Section A.1
`lifelines`	0.30.3	Nelson–Aalen slope -0.043	Section 3.4
`scipy`	1.17.1	Poisson likelihood of every hazard family; effort-dilution regression	Table 6
`numpy`	2.4.6	repository bootstrap draws (200, seed 0) behind every field interval	Section A.2
`pandas`	2.3.3	every aggregate, ratio and risk table	–
`anthropic`	0.120.2	maintainer, executor and judge 1 calls (`claude-haiku-4-5`); objectives (`claude-sonnet-5`)	Appendix B
`openai`	2.53.0	judge 2 (`gpt-5.6-luna`)	Section C.3
piecewise-exponential EM	ours	gamma frailty slopes and absorbed share	Table 7
DerSimonian–Laird	ours	two-stage combination of decay, frailty and forms	Table 6
penalized IRLS	ours	file-fixed-effect spline; 1,104 files	Section A.2
20 verifiers	ours	every satisfaction rate in Section 4	Table 10

REFERENCES

Worawalan Chatlatanagulchai, Hao Li, Yutaro Kashiwa, Brittany Reid, Kundjanasith Thonglek, Pattara Leelaprute, Arnon Rungsawang, Bundit Manaskasemsak, Bram Adams, Ahmed E. Hassan, and Hajimu Iida. Agent READMEs: An empirical study of context files for agentic coding, 2025. URL https://arxiv.org/abs/2511.12884. arXiv:2511.12884.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2025. doi: 10.3233/FAIA251160. URL https://arxiv.org/abs/2504.19413.

Beat Fluri, Michael Würsch, and Harald C. Gall. Do code and comments co-evolve? on the relation between source code and comment changes. In *Proceedings of the 14th Working Conference on Reverse Engineering (WCRE)*, pp. 70–79, 2007.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3 (4):128–135, 1999. doi: 10.1016/S1364-6613(99)01294-2.

Yingjie Gu, Wenjian Xiong, Liqiang Wang, Pengcheng Ren, Chao Li, Xiaojing Zhang, Yijuan Guo, Qi Sun, Jingyao Ma, and Shidang Shi. FSFM: A biologically-inspired framework for selective forgetting of agent memory, 2026. URL https://arxiv.org/abs/2604.20300. arXiv:2604.20300.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of ACL*, 2024. URL https://arxiv.org/abs/2310.20410. arXiv:2310.20410.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Donald E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.

Thomas D. LaToza, Gina Venolia, and Robert DeLine. Maintaining mental models: A study of developer work habits. In *Proceedings of the 28th International Conference on Software Engineering (ICSE)*, pp. 492–501, 2006. doi: 10.1145/1134285.1134355.

M. M. Lehman. Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68 (9):1060–1076, 1980.

Gili Lior, Asaf Yehudai, Ariel Gera, and Liat Ein-Dor. WildIFEval: Instruction following in the wild. In *Proceedings of the Generation, Evaluation and Metrics (GEM) Workshop at ACL 2026*, 2026. URL https://arxiv.org/abs/2503.06573. arXiv:2503.06573.

Jai Lal Lulla, Seyedmoein Mohsenimofidi, Matthias Galster, Jie M. Zhang, Sebastian Baltes, and Christoph Treude. On the impact of AGENTS.md files on the efficiency of AI coding agents. In *ICSE 2026 Journal-Ahead Workshop (JAWs)*, 2026. URL https://arxiv.org/abs/2601.20404. arXiv:2601.20404.

James G. March, Martin Schulz, and Xueguang Zhou. *The Dynamics of Rules: Change in Written Organizational Codes*. Stanford University Press, Stanford, CA, 2000.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pp. 109–165. Academic Press, 1989. doi: 10.1016/S0079-7421(08)60536-8.

Steve McConnell. *Code Complete: A Practical Handbook of Software Construction*. Microsoft Press, Redmond, WA, 2 edition, 2004.

Michael Nygard. Documenting architecture decisions. Blog post, 2011. URL https://cognitect.com/blog/2011/11/15/documenting-architecture-decisions.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems, 2023. URL https://arxiv.org/abs/2310.08560.

Sheena Panthaplackel, Pengyu Nie, Milos Gligoric, Junyi Jessy Li, and Raymond J. Mooney. Learning to update natural language comments based on code changes. In *Proceedings of ACL*, 2020. URL https://arxiv.org/abs/2004.12169. arXiv:2004.12169.

Sheena Panthaplackel, Junyi Jessy Li, Milos Gligoric, and Raymond J. Mooney. Deep just-in-time inconsistency detection between comments and source code. In *Proceedings of AAAI*, 2021. URL https://arxiv.org/abs/2010.01625. arXiv:2010.01625.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory, 2025. URL https://arxiv.org/abs/2501.13956.

Martin Schulz. Limits to bureaucratic growth: The density dependence of organizational rule births. *Administrative Science Quarterly*, 43(4):845–876, 1998.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31210–31227. PMLR, 2023.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. CODA-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of CVPR*, 2023.

Mahan Tafreshipour, Aaron Imani, Eric Huang, Eduardo Almeida, Thomas Zimmermann, and Iftekhar Ahmed. Prompting in the wild: An empirical study of prompt evolution in software repositories. In *Proceedings of the 22nd International Conference on Mining Software Repositories (MSR)*, 2025. URL https://arxiv.org/abs/2412.17298. arXiv:2412.17298.

Yingchen Tian, Yuxia Zhang, Klaas-Jan Stol, Lin Jiang, and Hui Liu. What makes a good commit message? In *Proceedings of the 44th International Conference on Software Engineering (ICSE)*, 2022. doi: 10.1145/3510003.3510205. URL https://arxiv.org/abs/2202.02974. arXiv:2202.02974.

James W. Vaupel, Kenneth G. Manton, and Eric Stallard. The impact of heterogeneity in individual frailty on the dynamics of mortality. *Demography*, 16(3):439–454, 1979.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. DualPrompt: Complementary prompting for rehearsal-free continual learning. In *Proceedings of ECCV*, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of CVPR*, 2022b.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911. IFEval; arXiv:2311.07911.

Xueguang Zhou. The dynamics of organizational rules. *American Journal of Sociology*, 98(5): 1134–1166, 1993.

LIMITATIONS

Section 3 is observational. The declining hazard survives sweeps of the rewrite fraction and the minimum file history (Table 9); we held three measurement choices fixed: the 70 fuzzy-match similarity, a matcher validated on 50 hand-annotated transitions (Appendix A.1), and a per-corpus segmentation grammar (Appendix A.1). Residual matcher error biases the age slope toward zero, against our finding. The corpus rejects fixed-effort dilution but cannot exclude partial dilution (Appendix A.2).

Section 4.2 isolates the mechanism in small worlds: covers of two or three instructions against a median file's 39, a horizon of 51 steps, one model family, and mechanically verifiable constraints. Three threats in real files remain unpriced: a rarely triggered instruction can look unneeded before its failure recurs, a comment records one cause for an instruction that may come to serve two, and an executor upgrade can leave a faithful comment defending an obsolete instruction.

Section 4.3 seeds 16 distractors into prompts carrying 4–5 human-written constraints, so it prices what excess costs and what prompt comments recover, not whether excess grows at that size. An arm-blind LLM judge scores its prose constraints, and the contrasts reproduce under a second judge (Appendix C.3).

Our scope is agent context files on public GitHub: 1,867 repositories carrying `CLAUDE.md`, `AGENTS.md`, or `copilot-instructions.md`. No result speaks to non-English instructions, system prompts or agent skill files.

A THE OBSERVATIONAL CORPUS

A.1 INSTRUCTION LIFETIMES FROM COMMIT HISTORY

Table 4 traces the corpus from sampling frame to estimation panel. The frame is the repository selection of Chatlatanagulchai et al. (2025): public GitHub repositories with a `CLAUDE.md`, `AGENTS.md` or `copilot-instructions.md`. We walk every version of every context file forward, so no stage conditions on survival. A three-stage cascade (exact, normalized, then fuzzy above 70) matches instructions across versions; an unmatched old instruction is a death, an unmatched new one a birth. Missed rewordings fall on old instructions and add hazard there, so the decline we report is a lower bound. A *rewrite* kills at least 75% of the live instructions in a file holding at least 5, and a *migration* moves text to a sibling file; both are censored as competing risks (Section A.3).

Table 5 lists the measurement gates. An *instruction* is one clause-level imperative, split on markdown structure and then sentence boundaries; headings, code, tables, bold labels and fragments under 2 words are *payload*, excluded from the risk set but counted in Section 3.1. Every reported ratio is invariant to a constant rescaling of $|D_t|$. The matcher scores 1.000 precision and 0.933 recall on 50 hand-annotated transitions (floor 0.85/0.80).

Table 4: What each pipeline stage keeps, from sampling frame to estimation panels. *Repos.*: repositories represented; N: count in the stage's unit. The death rows partition ended spells under the canonical rewrite rule (Section A.3); the two-stage row is the repositories with at least five deletions (Table 6); the collapsed panel is Figure 5's cut.

Stage	Rule	Repos.	N
repositories	≥ 1 tracked context file	1,867	1,867
multi-version files	≥ 2 versions	1,558	1,801
tracked transitions	consecutive version pairs	1,510	299,440
instruction spells	matched across transitions	1,856	247,694
deaths: deletion	the hazard's event	1,124	40,355
deaths: rewrite-censored	$\geq 75\%$ of live instructions killed, floor 5	640	84,233
deaths: migration-censored	text reappears in a sibling file	37	682
hazard population	file history ≥ 3 commits	1,095	177,995
two-stage fits	≥ 5 deletions per repository	430	–
collapsed panel	converges under every family, ≤ 4 half-lives	284	15,574

Table 5: Every measurement gate passes on the canonical corpus. Columns: the gate's threshold and this run's value. Gate B is measured on a 50-transition hand-annotated pool. *Grow:shrink*: share of multi-version repositories gaining instructions over share losing them; *tail share*: fraction of deletions at or above the median instruction age.

Gate	Criterion	Threshold	Observed	Pass
0	Median per-repo net $\Delta D > 0$	> 0	$+7$	✓
0	Grow:shrink ratio	> 1.5	2.42	✓
A	Deletions excluding rewrites	≥ 300	28,426	✓
A	Tail share at/above median age	$\geq 20\%$	57.0%	✓
B	Matcher precision / recall	≥ 0.85	1.000 / 0.933	✓
–	Median rel. growth, all multi-version files: count vs. length	count $>$ length	1.193 vs. 1.028	✓

A.2 Hazard families and rival mechanisms

Every estimator reads deletion risk against age in commits touching the file (Equation 3); spells leave the risk set by deletion, competing risk, or end of history. All fits share one population: spells whose history could reach 3 commits (177,995 lifetimes, 1,095 repositories). Three estimators agree on sign and shape: a Nelson–Aalen estimate smoothed at 2 commits over ages 0 to 50, the slope Section 3.4 quotes; a penalized spline with file fixed effects (1,104 files); and the parametric fits below. Intervals bootstrap over repositories (200 draws).

Table 6: The deletion hazard is geometric under both estimators; every bounded family fits worse. Each row fits $h(a)$ per repository by Poisson likelihood; *Repos.*: fits converging with finite variance. Rows rank by R^2_w, fit to repository-weighted per-age rates (<0: worse than their mean), as likelihoods sum over different repositories. *Bounded iff*: finite expected lifetime; *Decay*: its parameter (95% CI); τ^2: between-repository variance of log decay. Two-stage combines by DerSimonian–Laird over repositories with ≥ 5 deletions; hierarchical by maximum likelihood, no floor.

Estimator	Family	$h(a)$	Bounded iff	Repos.	τ^2	Decay	95% CI	R^2_w
two-stage (canonical)	**geometric**	$\boldsymbol{s\,e^{-ba}}$	**never**	**430**	**0.97**	$\boldsymbol{b = 0.222}$	$\mathbf{[0.199, 0.247]}$	**0.815**
	Weibull	$k\,a^{p-1}$	always	439	0.22	$p = 0.809$	$[0.762, 0.858]$	0.291
	constant	s	always	646	0.74	–	$[0.028, 0.032]$	-0.313
	gamma mixture	$s/(1+\theta sa)$	$\theta < 1$	423	0.71	$\theta = 7.034$	$[6.388, 7.747]$	-0.482
	Burr XII	$\alpha s(as)^{c-1}/(1+(as)^c)$	$\alpha > 1$	448	0.76	$\alpha = 0.151$	$[0.135, 0.168]$	-0.739
hierarchical	geometric	$s\,e^{-ba}$	never	517	0.94	$b = 0.241$	$[0.218, 0.267]$	0.658
	gamma mixture	$s/(1+\theta sa)$	$\theta < 1$	551	3.27	$\theta = 8.351$	$[7.051, 9.891]$	0.543
	Weibull	$k\,a^{p-1}$	always	551	0.23	$p = 0.813$	$[0.767, 0.863]$	0.082
	constant	s	always	921	1.28	–	$[0.022, 0.026]$	-0.730
	Burr XII	$\alpha s(as)^{c-1}/(1+(as)^c)$	$\alpha > 1$	595	1.52	$\alpha = 0.107$	$[0.094, 0.122]$	-1.489

Table 6 fixes the family. Five hazard forms, each answering whether Equation 4 converges, fit the per-(repository, age) risk table by Poisson likelihood. We fit each repository and combine by random effects, so between-repository spread enters as τ^2 rather than as curvature that would favour a bounded family.

Table 7: Within-repository frailty shrinks the age slope without reversing it. Both rows fit a piecewise-exponential hazard per repository (at least five deletions and two files; §A.2), the second with a gamma frailty shared across the repository's files, and combine slopes by DerSimonian–Laird. Columns: repositories, deletions, combined log-hazard slope per commit (95% CI), its τ^2, median frailty variance θ, and the share of slope the frailty absorbs. Deaths are censored at 50 commits.

Frailty	Repos.	Deaths	Slope	95% CI	τ^2	θ	Absorbed
no frailty	118	10,226	-0.0836	$[-0.1054, -0.0619]$	0.0098	–	–
shared within file	118	10,226	-0.0796	$[-0.1013, -0.0580]$	0.0096	0.001	4.8%

Table 7 tests whether heterogeneity, not duration dependence, drives the decline. A gamma frailty with variance θ gives $h_{\text{pop}}(a) = h_0(a)/(1 + \theta H_0(a))$, which declines even from a flat h_0, so we ask how much of the slope it absorbs: a piecewise-exponential EM fit within each of 118 repositories (10,226 deletions), with and without a frailty shared across the repository's files.

Effort dilution — fixed deletion effort per commit spread over a growing file — predicts that deletions scale with instructions at risk at exponent 0, against 1 under per-instruction risk. Poisson regression gives $\gamma = 0.74$ (0.74 ± 0.13) for deletions and 0.14 (0.14 ± 0.06) for additions, rejecting fixed effort.

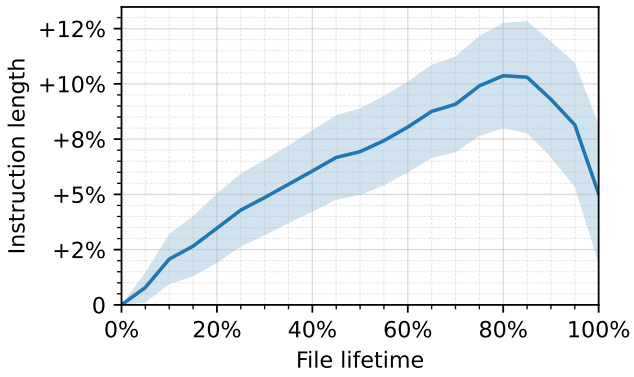


Figure 7: Instructions grow longer over a file's life, but far less than they grow numerous. Mean instruction length relative to the file's first version, $\bar{\ell}_t/\bar{\ell}_0 - 1$, against normalized lifetime, with a 95% band over 1,776 files. Length peaks at +10% against the count's +226% (Figure 1b); payload grows +140%.

Figure 7 separates count from elaboration. Instruction length peaks at +10% against the count's +226%, and payload grows +140%: the ratchet adds rules rather than lengthening them.

A.3 Sensitivity to the measurement thresholds

Rewrites and migrations censor 67.8% of deaths, and rewrite deaths outnumber hazard deletions 2.1 to one: maintainers replace rather than prune, as Equation 5 predicts. In Table 8, the rewrite is a mass deletion (a), its cut and regrowth hold at every event-study window (b), and growth after later rewrites is faster (c), so the ratchet is not self-limiting.

Table 9 moves one analysis choice at a time around the published configuration: the rewrite fraction at the canonical floor over 3 cells, and the minimum file history over 3 to 5 commits. The slope is negative with an interval excluding zero in all 5 cells, from -0.022 to -0.050 per commit, and does not track how much the rule censors: -0.043 at 66.1% censored, -0.050 at 85.5%, and shallower between. Geometric is the best form in 5 of 5 cells, leading by at least 0.280 in R^2_w with decay within

Table 8: The rewrite is a mass deletion, its shape does not depend on the window, and growth after it is no slower. **(a)** What rewrite commits do, corpus-wide. **(b)** The event study of Figure 4 at each window half-width w: median D_t at $-w$, 0 and $+w$ relative to the pre-rewrite version, over files observed at every offset, emptying events excluded; $\star$ marks the plotted width. **(c)** Growth within inter-rewrite segments (≥ 2 commits); segment k starts at the k-th rewrite. The rewrite threshold is fixed at 75%.

(a) Anatomy of a rewrite commit

Rewrite commits	1,077
Median instructions killed / born	52 / 12
Events leaving the file empty	30.6%
Events replacing $\geq$ half the dead text	33.0%
Events carrying any cross-file migration	3.3%

(b) Event study vs. window half-width

Window w	n	$k{=}-w$	$k{=}0$	$k{=}+w$
± 2	252	-2%	-60%	-48%
± 3	175	-7%	-64%	-51%
± 4	134	-9%	-65%	-53%
± 5	111	-11%	-70%	-52%
$\pm 10^{\star}$	52	-21%	-73%	-61%

(c) Growth within inter-rewrite segments

Segment	n	Median	Growing	Net ΔD
0	1,641	$\times 1.35$	77.9%	$+34.9$
1	454	$\times 1.40$	81.5%	$+37.2$
2	119	$\times 1.45$	80.7%	$+40.2$
3+	126	$\times 1.68$	85.7%	$+52.3$

Table 9: The deletion-hazard decline survives every choice varied here. Each block moves one choice off the published configuration (rewrite floor 5, fraction 0.75, minimum file history 3 commits, minimum age 1), whose row is **bold**. *Rewrite-censored*: share of deaths the rule removes; slope: $d \log h/da$ per commit with repository-bootstrap 95% CI; *Hazard form*: top two of 5 shapes by R^2_w under the two-stage estimator of Table 6. Rows share one death set and bootstrap seed.

				Hazard form (two-stage)			
		Rewrite-	Log-hazard slope	Best		Runner-up	
Setting	Deletions	censored	[95% CI]	form	R^2_w	form	R^2_w
Rewrite fraction of live instructions (floor 5)							
0.25	16,383	85.5%	-0.050 $[-0.062, -0.041]$	`geometric`	0.913	`weibull`	0.633
0.50	26,643	76.3%	-0.034 $[-0.044, -0.022]$	`geometric`	0.916	`weibull`	0.562
0.75	**38,138**	**66.1%**	$\mathbf{-0.043}$ $\mathbf{[-0.054, -0.032]}$	**`geometric`**	**0.815**	**`weibull`**	**0.291**
Minimum file history (commits)							
3	**38,138**	**66.1%**	$\mathbf{-0.043}$ $\mathbf{[-0.054, -0.032]}$	**`geometric`**	**0.815**	**`weibull`**	**0.291**
4	36,712	65.5%	-0.038 $[-0.055, -0.023]$	`geometric`	0.789	`weibull`	0.468
5	35,208	65.0%	-0.039 $[-0.056, -0.025]$	`geometric`	0.756	`weibull`	0.322

0.200 to 0.222. Post-rewrite growth outpaces pre-rewrite growth in 2 of the 3 rewrite cells, failing at the fraction that censors the most deaths.

B The Inverse-IFEval Testbed

B.1 Worlds, roles, and the censored feedback channel

A world is an IFEval item with at least two verifiers: 184 per seed, 552 histories per arm over 3 seeds, with covers of two or three. The unit of analysis is (seed, world), and a stronger model writes each world's objective o_j once. At each of T steps the harness samples 1 task and draws 2 constraints;

the executor answers under the current prompt with comments stripped; a fresh maintainer adds and deletes instructions, carrying only the prompt and what its arm persists. There is no rewording, so every instruction has an unambiguous age, as in Section 3.2. Both roles are `claude-haiku-4-5`.

Table 10: The hidden constraints and the only feedback the maintainer receives. Rows: verifier types, grouped by IFEval family; columns: the (seed, world) pairs containing each type, and its failure complaint. A complaint names a dimension, never a parameter or threshold: the censoring §2 assumes. Counts exceed the world count because each world carries two or three constraints.

Family	Verifier type	Worlds	Complaint returned on failure
keywords	`existence`	69	the response is missing some required content
	`forbidden_words`	81	the response contains wording it must not use
	`frequency`	75	certain wording is not used the right number of times
	`letter_frequency`	57	a certain letter does not appear the right number of times
length	`number_paragraphs`	48	the response is not split into the right number of paragraphs
	`number_sentences`	99	the response has the wrong number of sentences
	`number_words`	108	the response's word count is off
content	`number_placeholders`	42	the response needs a different number of bracketed placeholders
	`postscript`	36	the response is missing a postscript
format	`json_format`	27	the response is not valid JSON
	`multiple_sections`	24	the response is not organized into the expected sections
	`number_bullet_lists`	54	the response does not have the right number of bullet lists
	`number_highlighted_sections`	87	the response needs a different number of highlighted sections
	`title`	36	the response is missing a properly formatted title
case	`capital_word_frequency`	39	the number of fully-capitalized words is off
	`english_capital`	42	the response's letter casing is wrong
	`english_lowercase`	75	the response's letter casing is wrong
punctuation	`no_comma`	87	the response's punctuation breaks a rule
start / end	`end_checker`	27	the response does not end the required way
	`quotation`	69	the response does not begin and end the required way

Table 10 lists each constraint type and its complaint, which names a dimension and returns a bare verdict on a pass: the censoring Section 2 assumes. The arms separate only under vague verdicts, censored passes, withheld identifiers, an objective silent on brevity, and a horizon of at least ten steps. Each condition removes information a real maintainer also lacks; disclosing the verifier specification settles the uncommented arm at +3.0% excess.

Figures 8–10 give the maintainer's full instructions and one real prompt in both projections. An instruction tells the executor what, never why; a comment, `[di] text # comment`, addresses the next maintainer. It is written once at addition, never edited, and capped at 1,024 characters, so a rationale without a live instruction has nowhere to live, as Equation 5 formalizes. The harness strips comments before the executor reads the prompt and rejects instructions citing past failures; Section B.4 lifts the strip.

B.2 Cover attainment by maintainer tier

The replicated design ran $T{=}15$ at 3 seeds (Section 4.2 reports $T{=}51$). Commented prompts end at -5.8% excess against +60.4% uncommented and +53.2% under comment-shaped noise (Table 13), at a satisfaction difference of -0.8pp (-0.8 ± 2.1pp), within the $\pm$5pp tolerance.

Maintainer system prompt (all arms)

```
You maintain a reusable prompt (a numbered list of directives) for an assistant that
  repeatedly performs variations of one task. After each round, some responses were
  scored against hidden requirements; you see this round's scores below.

Optimize the prompt for LONG-RUN average score. You may add directives and delete
  existing ones.

Each response is checked against a random sample of the hidden requirements. You are NOT
  told how many requirements exist, which ones were sampled, or which directive (if any)
  satisfied a passing one; a failure may include a description of what went wrong.

Rules for directives:
- A directive is one imperative sentence addressed to the assistant about what the
  RESPONSE must do. It must NEVER mention scores, failures, rounds, or history.
- You cannot reword a directive; to change one, delete it and add a new one.
- With each add, supply `comment` (a note to the future maintainer: which observed
  failure prompted this, in your own words, e.g. "task 1 was flagged for comma use") and
  `cause` (one short phrase naming the requirement you believe it targets, e.g. "no
  commas allowed").

Return only the operations. An empty ops list keeps the prompt unchanged.
```

Figure 8: The maintainer's system prompt, identical across arms; only the addendum (Figure 9) differs. Figure 10 shows what each role sees. Line breaks are set for this page.

Protocol addendum (latent-reasoning comment arm only)

```
Your ONLY memory across rounds is the `comment` attached to each directive (at most
  1,024 characters, frozen when the directive is added). Write every comment as a commit
  message to all future maintainers, containing:
- "r<N>:" — this round's number, inferred as one more than the highest r<N> in the
  existing comments (r1 if there are none);
- the failure description that prompted this directive, quoted, and how many rounds it
  has been recurring;
- the lineage of this failure: which approaches were already tried against it and
  FALSIFIED (copy forward any falsifications recorded in the comments of directives you
  are deleting now — a deleted directive's comment vanishes with it);
- if this directive replaces a deleted one that might still have been doing work, the
  deleted directive's exact text in quotes, so a successor can restore it verbatim.
Before deleting any directive, read its comment: if its failure has not recurred since
  the round it was added, the directive is likely PREVENTING that failure — keep it.
  Delete a directive when its comment's failure kept recurring anyway (falsified), or
  when it duplicates another directive's target.
```

Figure 9: The protocol addendum, appended to Figure 8 for the latent-reasoning comment arm only; it is the sole input difference between arms. Line breaks are set for this page.

Table 2 varies the maintainer alone: the uncommented arm's excess and the protocol's advantage both grow with capability. The Sonnet 5 maintainer overran the comment budget on two of five additions, so its row records compliance rather than capability.

B.3 Ablations and Failed Protocols

Table 11 tests whether the effect is the comment's form or content and which COMMIT field carries the pruning (field names as in Figure 10(a)); each row is compared with its own uncommented control. Table 12 lists every earlier protocol and how it failed.

(a) The maintained prompt as one document (objective, instructions, comments)

```
Objective: Plan a grocery store trip in Lima to buy local goods.

[d17] Ensure the response is complete and untruncated, with every section fully written
  out.  # r2: "the response's letter casing is wrong" and "the response is missing some
  required content" — recurring since r1. d16 (merged directive) failed to prevent
  truncation. Deleting d16; adding two separate directives to address completeness and
  casing independently.
[d19] Maintain proper capitalization throughout: use sentence case for regular text,
  capitalize the first letter of proper nouns and the first word of sentences, and avoid
  unnecessary ALL CAPS except for acronyms.  # r3: "the response's letter casing is
  wrong" — recurring since r1. Previous directive d18 was too vague. Replaced with more
  specific guidance on capitalization rules to address inconsistent casing patterns.
[d20] Provide the complete response without truncation; do not cut off content
  mid-sentence or mid-section, and ensure all promised sections and details are fully
  included.  # r3: "the response is missing some required content" — recurring since r1.
  Previous d17 mentioned truncation but d16 failed to prevent it. Response is being cut
  off mid-sentence. Need explicit requirement to never truncate output and complete all
  sections fully.
```

(b) What the maintainer is handed each round

```
## Current prompt directives
(panel (a)'s commented list, verbatim)

## This round's scores
task 0: a requirement (the response's letter casing is wrong) FAILED
task 0: a requirement (the response is missing some required content) FAILED
```

(c) What the executor is handed: the same instructions, comments stripped

```
Follow ALL of these directives in your response:
1. Ensure the response is complete and untruncated, with every section fully written
  out.
2. Maintain proper capitalization throughout: use sentence case for regular text,
  capitalize the first letter of proper nouns and the first word of sentences, and avoid
  unnecessary ALL CAPS except for acronyms.
3. Provide the complete response without truncation; do not cut off content mid-sentence
  or mid-section, and ensure all promised sections and details are fully included.

--- user message ---

(panel (a)'s objective, verbatim)
```

(d) Placebo comment pool

```
- added after one of the responses was flagged in an earlier round
- a response failed a requirement around this in a previous round
- several earlier rounds seemed to have trouble with this
- this came up in the scores a while back
- added to address a recurring issue seen earlier
- an earlier round's feedback pointed at something like this
- one of the tasks was marked down for this before
- kept seeing failures that looked related to this
```

Figure 10: One real maintained prompt and the two projections the roles receive: the protocol arm's last round on world 159 of seed 0, at 3 instructions (arm median 2). **(a)** The maintainer's artifact: objective, instructions and comments, seen by no role. **(b)** That list with a censored report. **(c)** The executor's call as the render function produces it: comments and identifiers stripped, objective in the user message. **(d)** The placebo arm's comment pool. Notation follows Appendix B.1.

Table 11: A comment must carry outcomes and reasoning lineage. Each row compares a run's arm with its own control. Comment-shaped noise has the comment arm's form and the control's information; each clause row drops one COMMIT field. z: paired per-world statistic on Δ. Clause rows use n=64 worlds and one seed.

	Excess size		Satisfaction	
Ablation	Δ (%)	z	Δ (%)	z
comment-shaped noise	-3.1	-0.33	$+3.3$	0.77
narrative without outcomes	$+8.8$	0.86	-3.3	-0.82
COMMIT, full schema	-48.9	-2.78	$+4.1$	0.50
– round counter	-47.4	-2.63	-1.9	-0.23
– recurrence count	-30.9	-1.83	-3.2	-0.42
– reasoning lineage	-34.3	-2.14	-9.6	-1.19
– verbatim restore quote	-43.5	-2.43	-8.9	-1.09

Table 12: The failed protocols and what each persisted, in the order tried, each from its own run. Columns: excess size $\frac{|D_T|}{|D_\star|}-1$ for the protocol arm and its run's uncommented control, last-3-step satisfaction, and per-world calibration (Table 14). Compare within rows only, since runs differ in worlds drawn (N). FREEFORM narratives exceed their control; persisting actions without outcomes is worse than persisting nothing; note formats plateau; WINDOW reaches cover, but a party other than the maintainer writes its digest, so we do not deploy it.

	Persists	Failure mode	N	Excess	(control)	Sat.	Empty	|dev|
FREEFORM	the strategy tried, unstructured	records actions without their outcomes, so a successor inherits a legitimized failing hypothesis and extends it	184	$+70.0\%$	$+56.2\%$	0.377	2.7%	0.934
LEDGER	a confirmed / falsified / open note	suppresses additions but never prunes: a compressed note carries the falsification record but not per-instruction non-recurrence	184	$+42.8\%$	$+56.4\%$	0.407	4.9%	0.712
TALLY	one evidence line per live instruction	adds nothing over the note, consistent with a bounded note saturating rather than its format being wrong	64	$+57.8\%$	$+84.1\%$	0.411	3.1%	0.792
WINDOW	the note plus a retained 5-round digest of scores and edits	settles at cover on the arm mean, but empties a fifth of prompts to get there; the digest is authored by a party other than the maintainer, so it is not a comment protocol	184	-1.9%	$+56.4\%$	0.361	17.9%	0.590
CALIBRATED	the digest plus observed / guess / evidence per comment	hedging suppresses addition globally and costs satisfaction	64	-4.7%	$+84.1\%$	0.333	14.1%	0.536

Table 13: The ratchet and its reversal replicate on every world draw. Upper block: final excess size $\frac{|D_T|}{|D_\star|} - 1$ per arm and seed, pooled, and across-seed SD in pp; each seed is an independent permutation sample of IFEval. Lower block: last-3-step satisfaction.

Excess size (%)	s0	s1	s2	Pooled	sd
no prompt comments	$+59.9$	$+49.8$	$+71.6$	$+60.4$	10.9
comment-shaped noise	$+54.9$	$+46.0$	$+58.8$	$+53.2$	6.5
comments w/latent reasoning	-0.1	-2.1	-15.1	-5.8	8.2
Satisfaction (%)	**s0**	**s1**	**s2**	**Pooled**	**sd**
no prompt comments	38.8	38.3	40.0	39.0	0.9
comment-shaped noise	40.0	40.0	40.8	40.3	0.4
comments w/latent reasoning	38.9	36.8	39.0	38.2	1.3

B.4 World-draw replication and the over-pruning cost

Table 13 shows the ratchet and its reversal on each of three independent world draws. The uncommented arm's across-seed spread of 10.9pp is small against the 66.2pp effect.

Table 14: The commented arm is closest to cover on the mean and per prompt. **(a)** Final-step calibration: mean $|D_T|/|D_\star|$, mean $\big||D_T|/|D_\star| - 1\big|$, and shares of worlds ending empty, within a quarter of cover, and above $1.5\times$ cover. The protocol empties one prompt in eight, against one in forty uncommented. **(b)** Last-3-step satisfaction per arm, split by whether the protocol ended empty, with the paired difference from the protocol (95% bootstrap CI). Parity holds where the protocol kept instructions and fails on the hard worlds it emptied.

(a) Per-world calibration to the minimum cover

Arm	N	Mean ratio	Mean $\lvert\text{ratio} - 1\rvert$	Empty	Within $\frac{1}{4}$	Above $1.5\times$
no prompt comments	552	1.604	0.921	2.4%	33.3%	22.3%
comment-shaped noise	552	1.532	0.822	1.1%	35.0%	21.9%
comments w/latent reasoning	552	0.942	0.481	12.1%	37.3%	9.8%

(b) Satisfaction on the worlds the protocol emptied, and on the rest

Arm	N	Satisfaction	95% CI	Paired Δ, protocol $-$ arm	95% CI
worlds the protocol emptied					
no prompt comments	67	0.279	$[0.209, 0.348]$	-0.020	$[-0.065, +0.025]$
comment-shaped noise	67	0.328	$[0.256, 0.400]$	-0.070	$[-0.114, -0.027]$
comments w/latent reasoning	67	0.259	$[0.199, 0.321]$	—	
worlds it kept instructions on					
no prompt comments	485	0.406	$[0.377, 0.436]$	-0.007	$[-0.031, +0.016]$
comment-shaped noise	485	0.413	$[0.383, 0.443]$	-0.014	$[-0.034, +0.007]$
comments w/latent reasoning	485	0.399	$[0.368, 0.429]$	—	

Table 14 prices over-pruning. The protocol reaches cover per world (a) but empties one prompt in eight. Splitting worlds by that outcome (b), satisfaction parity holds where the protocol kept instructions and fails on the emptied worlds, which are hard for every arm; the deficit bounds the cost of deletion. Deployments should set a pruning floor.

Rendering the protocol's comments into the executor's prompt too (one seed, 85.1% of executor calls exposed) ends at +28.5% excess against -0.1% over 184 worlds (28.6pp, 28.6 ± 26.5pp) at a satisfaction difference of -1.7pp (-1.7 ± 3.4pp): inline rationale costs size and buys no correctness. Maintenance vocabulary appears in 0.04% of exposed completions against 0.00% under the strip, so the strip isolates the comment channel rather than preventing leakage.

Table 15: The arm effect lives where the distractors are. Every design cell of §4.3 at the final round: N hidden constraints, K withheld from the seeded prompt, D distractors. SSR: per-constraint satisfaction over the provided $N-K$ (interference), the withheld K (recovery), and all N; $|D_t|$ and $|D_T|$: the prompt the executor saw and the prompt after the last round; trunc.: share of responses hitting the token cap. At D=16 TREATMENT holds all-N SSR at its D=0 level while CONTROL falls. Size columns are descriptive: the seeded comment states its own falsification.

					SSR					
N	K	D	Arm	n	provided	withheld	all	$\lvert D_t\rvert$	$\lvert D_T\rvert$	trunc.
5	1	0	CONTROL	32	0.547	0.531	0.544	5.500	6.594	0.219
5	1	0	PLACEBO	32	0.578	0.500	0.562	5.250	5.781	0.188
5	1	0	TREATMENT	32	0.633	0.531	0.613	5.750	6.250	0.281
5	1	16	CONTROL	32	0.531	0.438	0.512	14.750	14.406	0.312
5	1	16	PLACEBO	32	0.523	0.500	0.519	14.406	12.969	0.312
5	1	16	TREATMENT	32	0.656	0.500	0.625	6.500	7.031	0.250
6	1	0	CONTROL	32	0.688	0.531	0.661	6.656	6.969	0.375
6	1	0	PLACEBO	32	0.650	0.500	0.625	6.312	6.906	0.344
6	1	0	TREATMENT	32	0.575	0.500	0.562	6.562	6.594	0.438
6	1	16	CONTROL	32	0.525	0.344	0.495	14.812	14.844	0.344
6	1	16	PLACEBO	32	0.569	0.406	0.542	14.969	13.688	0.344
6	1	16	TREATMENT	32	0.631	0.531	0.615	7.125	7.562	0.406

C THE WILDIFEVAL REPLICATION

C.1 THE DESIGN GRID AND THE ARM-BLIND JUDGE

Table 15 reports every cell behind Section 4.3's four contrasts. WildIFEval has human-written constraints and no code verifiers, so `claude-haiku-4-5` judges one constraint against one response, seeing no prompt, arm or history; judge error is common to both arms, not differential.

C.2 INTERFERENCE AND RECOVERY

The effect is recovery from seeded noise. At $D = 0$ the arms do not separate consistently; at $D = 16$ the commented arm holds satisfaction while the control falls in both strata (Table 15).

C.3 JUDGE ROBUSTNESS AND AGREEMENT

Table 16: Every contrast of §4.3 survives an independent judge. Rows: the contrasts of Appendix C.1. Columns 2–3: the paired TREATMENT−CONTROL difference under each judge (PLACEBO−CONTROL for placebo rows; D=16 vs. D=0 within CONTROL for interference), with 95% bootstrap interval over W=64 worlds; **bold**: interval excludes zero. `gpt-5.6-luna`, a same-tier judge from another vendor, re-scores `claude-haiku-4-5`'s stored responses with everything but the model fixed. Column 4: the per-world difference-in-differences between judges, which covers zero on every row. The size rows count directives no judge sees, so their DID is 0 by construction.

	Contrast under each judge		DID
Criterion	`claude-haiku-4-5`	`gpt-5.6-luna`	`claude-haiku-4-5` − `gpt-5.6-luna`
(i) interference	**−0.241** [−0.334, −0.149]	**−0.197** [−0.287, −0.108]	—
(ii) comments buy IF	**+0.116** [+0.051, +0.183]	**+0.078** [+0.006, +0.149]	+0.038 [−0.019, +0.096]
(v) not just recovery	**+0.116** [+0.048, +0.185]	**+0.083** [+0.005, +0.159]	+0.033 [−0.031, +0.095]
(aux) recovery	**+0.125** [+0.016, +0.250]	+0.062 [−0.062, +0.188]	+0.062 [−0.062, +0.203]
(iii) comments buy size	**−7.328** [−8.641, −5.953]	**−7.328** [−8.641, −5.953]	+0.000 [+0.000, +0.000]
(iv) placebo null (SSR)	+0.027 [−0.044, +0.101]	+0.033 [−0.036, +0.101]	−0.006 [−0.070, +0.061]
(iv) placebo null (size)	−1.297 [−2.656, +0.031]	−1.297 [−2.656, +0.031]	+0.000 [+0.000, +0.000]

Table 17: Judge agreement is substantial and constant across arms, so judge noise cannot have produced the arm gap in Table 16. Judge a = `claude-haiku-4-5` against b = `gpt-5.6-luna` on all 6,336 verdicts, pooled and per arm: raw agreement, Cohen's κ (95% bootstrap CI), each judge's pass rate, and lenience (b only $-$ a only)$/n$. κ spans 0.631–0.653 across arms; the judges disagree on 17.9% of verdicts, yet their pass rates differ by 0.002.

				Pass rate		
Slice	n	Agree	κ [95% CI]	`claude-haiku-4-5`	`gpt-5.6-luna`	Lenience
all arms	6,336	0.821	0.639 [0.620, 0.657]	0.545	0.547	+0.002
CONTROL	2,112	0.816	0.631 [0.598, 0.664]	0.539	0.534	−0.006
TREATMENT	2,112	0.819	0.632 [0.598, 0.665]	0.565	0.563	−0.002
PLACEBO	2,112	0.827	0.653 [0.618, 0.684]	0.529	0.543	+0.014

Table 16 re-scores all 6,336 verdicts under `gpt-5.6-luna`, a same-tier judge from another vendor, holding prompt, schema and decoding fixed. Every contrast but recovery keeps its sign and an interval excluding zero; the placebo stays null. The judges' difference-in-differences is 3.8pp ($+3.8 \pm 5.8$pp): no detectable judge difference in effect size.

Table 17 shows 82.1% verdict agreement at $\kappa = 0.639$ (0.639 ± 0.018), constant across arms, so judge noise cannot produce the arm gap.