IS CROSS-CONTEXT KV CACHE REUSE ALWAYS RELIABLE AND HARMLESS IN AGENT EXECUTION?

Anonymous authors
Paper under double-blind review

ABSTRACT

Cross-context KV cache reuse can reduce repeated prefill for reusable agent context, such as skill documents, tool schemas, and policy instructions. However, preserving task success does not necessarily preserve agent execution: KV reused from a different context can substantially alter an agent's responses and actions, while incurring additional execution costs or workspace differences even when the task is completed successfully. This paper explores the mechanism of cross-context KV cache reuse. We trace KV cache reuse from cache differences within model to agent behavior and task outcomes. Further, we characterize how reuse induces behavioral divergence, explain how divergent executions can still preserve task success, and derive a criterion for determining when such reuse remains permissible under execution requirements. Overall, this paper identifies and theorizes the impact of cross-context KV cache reuse on agent execution, guiding the reliable deployment of cross-context KV cache reuse in agent systems.

1 INTRODUCTION

As large language models (LLMs) scale and agent applications expand, inference cost becomes a major bottleneck. During each model invocation, the model computes the keys and values (KV) in each attention layer based on the input context. These KV are stored as the KV cache. Prefix caching reuses the KV cache when a request shares an identical prefix with an earlier one, avoiding repeated computation. LLM agents present a different pattern. Their requests repeatedly include the same text segments, such as skill documents, tool schemas, and policy instructions, but these segments appear after a conversation history that changes from turn to turn. Since these segments sit outside the prefix, cross-context reuse (Yao et al., 2025; Hu et al., 2025; Liu et al., 2026b) allows a cached segment to be reused even when the text before it differs. However, how this cross-context reuse affects the agent's execution and its task outcome remains unexplored.

When the KV cache of a text segment is reused across contexts, the KV inside that cache depend on the preceding context as well as the text itself. Under cross-context reuse, an LLM-based agent system reuses the same text segment while its conversation history, task state, and previous actions change. The KV cache of that segment was computed under the earlier context, so its KV can differ from those computed for the current request. We call a segment's cache a *stale KV cache* when it was computed under a different preceding context. *Stale KV cache reuse* uses that cache for the current request. Such reuse preserves the visible prompt while allowing its KV cache to differ.

We construct controlled tasks to isolate the effects of stale KV cache reuse. Experiments demonstrate that, even when all controlled tasks are completed successfully, the agent's responses and actions still change in most cases. Such behavioral divergence can alter execution costs, workspace states, or required actions, which are not captured by task success alone. Explaining this phenomenon requires understanding both how a difference in KV changes the agent's behavior and why the task outcome nevertheless survives. Existing studies approaching this issue can be grouped into two categories: 1) enabling reuse and validating outcomes, and 2) studying behavioral divergence and recovery. The former establishes mechanisms to reuse KV cache across requests (Kwon et al., 2023; Zheng et al., 2024; Yao et al., 2025; Hu et al., 2025; Liu et al., 2026b; Ye et al., 2025). These approaches significantly reduce prefill overhead while maintaining output quality. However, they check only the final output. Behavioral changes during execution stay unexplored.

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.

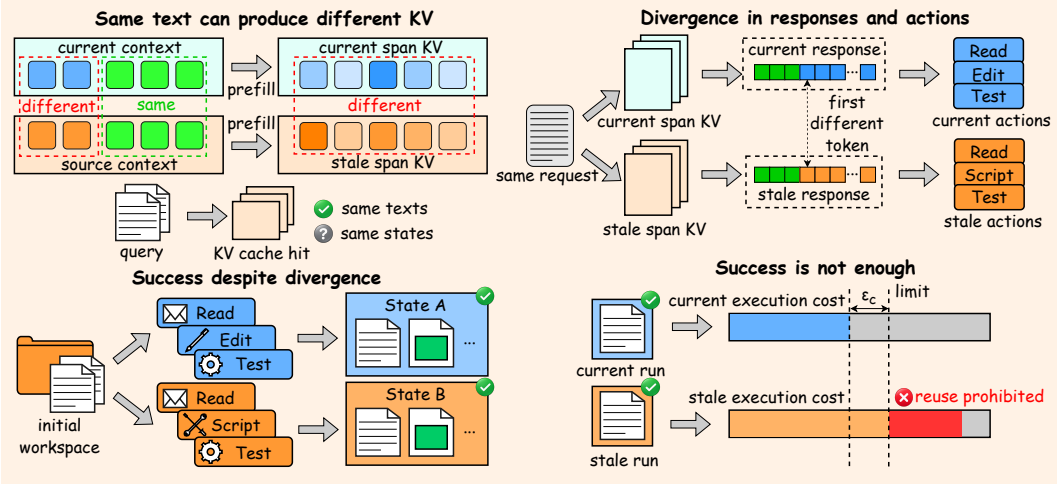

Figure 1: **The same visible input can produce different execution paths.** A cache hit matches the reused text but not its preceding context, so current (blue) and stale (orange) span KV differ. The responses split at the first different token and issue different actions. Both pass the task check, yet a cost increase beyond its budget prohibits reuse.

On the other hand, some studies explore how the agent's behavior diverges and how it recovers. Bigelow et al. (2025) show that replacing a single token can steer autoregressive generation onto a different path. Reflexion (Shinn et al., 2023) and CRITIC (Gou et al., 2024) show that external feedback repairs agent errors, and Huang et al. (2024) show that without such feedback the repair largely fails. Liang et al. (2026) observe that cache reuse can leave a judge's accuracy stable while changing its selected candidate. Although these studies effectively characterize divergence and recovery as separate phenomena, the link from stale KV cache reuse to divergence and then to task success remains unexplained.

To explain how cross-context KV cache reuse affects agent execution, we trace how stale KV cache reuse redirects agent behavior through internal KV differences, and how the resulting divergence is absorbed or propagates to the task outcome. **Inside the model**, the KV difference shifts the scores of candidate tokens. The top candidate leads each other candidate by a score gap, which we call the *margin*. When the shift lets another candidate close this gap and overtake the top candidate, the model emits that candidate. The changed token and the retained stale KV carry the change into the rest of the response. We call this change in the generated response *divergence*. **At the agent level**, a divergent execution can still succeed through alternative action sequences or environment feedback (Figure 1). We call such success *absorption*. Without either route, the task fails. Even absorbed executions can differ in cost, workspace state, and action compliance. Overall, stale KV cache reuse changes the agent's actions in ways that task success does not reflect.

This study explores how stale KV cache reuse alters agent behavior and when it remains permissible. To compare current and stale KV under identical inputs, we conduct controlled experiments on an OpenHands agent served by a vLLM engine that we extend with KV injection. The results show that 1) divergence depends on how the KV difference shifts relative scores against the margin, 2) smaller margins make divergence more frequent, and 3) the direction of the difference decides the task outcome. From these findings, we derive the conditions for permissible reuse: the stale execution must succeed and keep its costs, workspace changes, and actions within pre-specified requirements. In summary, this paper offers the following three insights:

- **A stale KV cache redirects the agent through token decisions.** The stale KV cache shifts the relative scores of candidate tokens and changes a token when this shift exceeds the margin. Divergence therefore starts at near-tied decisions. The changed token and the retained stale KV then shape later decisions, and the direction of the KV difference decides the task outcome.

- **Absorption preserves success along different execution paths.** Absorption, i.e., task success despite divergence, follows valid alternative paths, and removing corrective feedback affects current and stale executions alike. Further, success can still incur execution costs or workspace differences that the task check does not capture.
- **Permissible reuse depends on execution requirements, not on task success alone.** An absorbed execution can still cost more or leave a different workspace. Our criterion checks these costs and workspace changes to decide when stale KV cache reuse is permissible, and separates pairs that task success alone treats as equal.

2 Related Work

KV cache reuse. Reuse systems reduce prefill cost through prefix sharing (Kwon et al., 2023; Zheng et al., 2024), non-prefix splicing (Gim et al., 2024; Yao et al., 2025; Hu et al., 2025; Liu et al., 2026b; Agarwal et al., 2025), and cross-agent communication (Ye et al., 2025). Eviction, compression, and streaming reduce cache storage and transfer (Zhang et al., 2023; Li et al., 2024; Liu et al., 2024), while parallel and block-wise encoding process independent contexts (Yang et al., 2025b; Ma et al., 2025). Long-context benchmarks evaluate output quality after reuse (Li et al., 2025), yet unchanged accuracy can hide changed individual answers (Dutta et al., 2024). Whether the reused cache alters the agent's behavior, beyond what the output reveals, remains unexplored.

Divergence and recovery. A single token can redirect generation (Bigelow et al., 2025), and early mistakes can persist across turns (Laban et al., 2026). LLMs and agents recover through self-feedback (Madaan et al., 2023; Shinn et al., 2023) and tool critique (Gou et al., 2024), with external signals affecting recovery (Huang et al., 2024). AgenTracer attributes failure to execution steps (Zhang et al., 2026), and process metrics show progress differences that success rates hide (Ma et al., 2024). KV reuse can change a judge's selected candidate despite stable accuracy (Liang et al., 2026). These studies address divergence and recovery separately, without tracing how a cache-induced change resolves into task success or failure.

Internal representations. Causal tracing, activation patching, and circuit discovery localize effects within model computations (Meng et al., 2022; Zhang & Nanda, 2024; Marks et al., 2025). Probes decode represented information (Zou et al., 2023; Feng et al., 2025), while attention sinks (Xiao et al., 2024), off-distribution interventions (Grant et al., 2026), and numerical near-tie sensitivity (Yuan et al., 2025) clarify how internal changes affect outputs. How such internal changes affect downstream generation, agent actions, and task outcomes is not traced.

3 Preliminaries

To isolate the effect of KV cache reuse, we compare current and stale caches for the same text. For a reused span s, the current cache $\mathcal{K}_s^I$ is computed under the request's own preceding context, and the stale cache $\mathcal{K}_s^S$ under a different preceding context, which we call the source context. Both caches hold the same span tokens, and the stale keys are rotated with the model's positional encoding (Su et al., 2024) to the span's current positions. Stale KV cache reuse substitutes $\mathcal{K}_s^S$ for $\mathcal{K}_s^I$, the full reuse that non-prefix reuse systems start from (Yao et al., 2025; Hu et al., 2025), and both caches are loaded through the same procedure. We write Δ for stale-minus-current differences, so the cache difference is $\Delta\mathcal{K}_s = \mathcal{K}_s^S - \mathcal{K}_s^I$. Recomputation, a full prefill without a loaded cache, is a baseline.

The setting z fixes the model, the type of reusable text, the environment, and the feedback the agent receives. A *pair* runs one task of a setting twice, once with current KV and once with stale KV. For $a \in \{I, S\}$, the trajectory τ^a records the complete execution, and $Y_a \in \{0, 1\}$ indicates task success. The theoretical results assume shared inputs and deterministic decoding, environment transitions, and stopping rules (Appendix A). Probabilities $P(\cdot \mid z)$ average over the pairs of setting z (Appendix A.2).

Let $D_{\mathrm{tok}} = 1$ indicate that the two executions of a pair generate different responses at some turn, and $D_{\mathrm{act}} = 1$ that they produce different parsed actions. A fixed parser makes action divergence

imply response divergence, and we use $D = D_{\text{tok}}$ as the divergence indicator. When the responses diverge ($D = 1$) and both executions succeed ($Y_I = Y_S = 1$), we call the outcome *absorption*.

4 How Does Divergence Arise and Persist?

To explain how divergence arises and persists, we connect attention differences to token selection and subsequent agent interaction (Figure 2). We compare Qwen3-14B-FP8 (Yang et al., 2025a) and Ministral-3-14B-FP8 (Liu et al., 2026a), hereafter Qwen and Ministral, on 72 controlled tasks. Both use greedy decoding in OpenHands (Wang et al., 2025) on vLLM (Kwon et al., 2023). The tasks require file edits following the reused text (Appendix B). Two seeds per task give 144 pairs per model, with the same divergence and reuse assessment under both seeds. Responses diverge in 140 Qwen and all 144 Ministral pairs, yet every pair succeeds under both caches (Appendix Table 4).

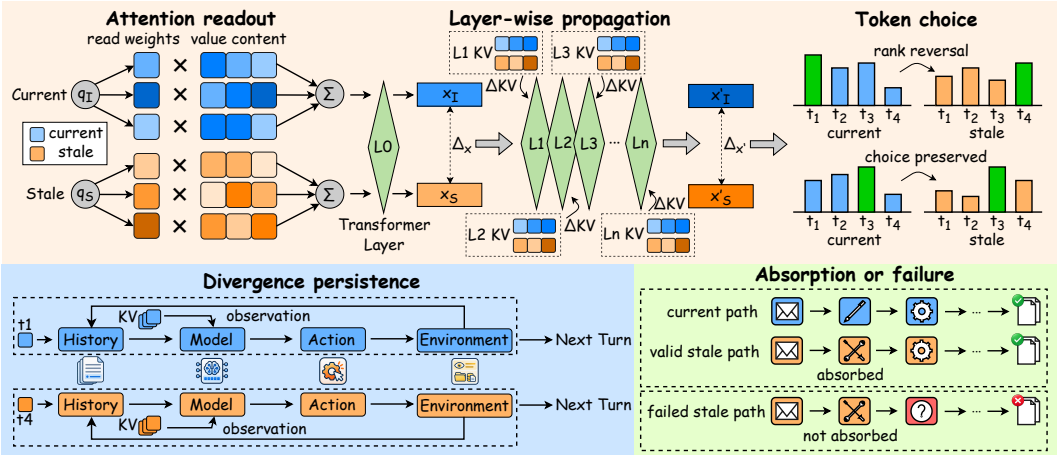

Figure 2: **From internal differences to execution outcomes.** Top: stale KV changes attention weights and values (Equation 1), each layer adds its own KV difference (Equation 2), and the score change reverses or preserves the token choice (Proposition 1). Bottom: the changed token, the retained KV, and the environment carry divergence into later turns (Equation 7), where a valid stale path is absorbed and a failed one is not. Labels x_I, x_S denote x_l^I, x_l^S, primes mark the final layer, q_I, q_S are queries, and t_i are candidate tokens, with green bars marking the selected token. In the bottom rows, t_1 and t_4 are the tokens selected under current and stale KV.

How stale KV changes attention. Attention outputs a weighted sum of value vectors (Vaswani et al., 2017). At layer l and query position i, let $\beta_{l,ij}$ weight value $v_{l,j}$ at attended position $j \le i$. For identical tokens, the output difference separates changed weights from changed values (Kobayashi et al., 2020):

$$\Delta o_{l,i} = \underbrace{\sum_{j \le i} \Delta\beta_{l,ij} v_{l,j}^I}_{\text{changed attention weights}} + \underbrace{\sum_{j \le i} \beta_{l,ij}^S \Delta v_{l,j}}_{\text{changed values}} . \tag{1}$$

Here $\Delta v_{l,j} = v_{l,j}^S - v_{l,j}^I$ includes differences both within and after the reused span (Appendix A.1).

From attention differences to token scores. Each layer passes the differences it receives to the next layer and adds a contribution from its own reused KV. For a shared token sequence, let x_l stack the layer-l hidden representations of all positions after the reused span. The shared tokens and positions give identical input embeddings, so $\Delta x_0 = 0$. Let $\mathcal{K}_l^a = (K_l^a, V_l^a)$ denote the layer-l keys and values of the span cache $\mathcal{K}_s^a$. Let F_l be layer l in exact arithmetic, a differentiable map from x_l and $\mathcal{K}_l$ to x_{l+1}. Expanding around the current execution splits the difference into three terms

$$\Delta x_{l+1} = \underbrace{J_l^x \Delta x_l}_{\text{propagated difference}} + \underbrace{J_l^{\mathcal{K}} \Delta \mathcal{K}_l}_{\text{layer KV contribution}} + \eta_l, \tag{2}$$

where J_l^x and $J_l^{\mathcal{K}}$ are the Jacobians of F_l with respect to the hidden representations and the reused KV. The remainder η_l collects nonlinear and finite-precision effects (Appendix A.1). Standard dot-product self-attention is not globally Lipschitz (Kim et al., 2021; Castin et al., 2024), so the token-level results below use the realized logit change.

After L layers, the output map ψ applies final normalization and the vocabulary projection to the entry $x_{L,i}$ of the final representation x_L at position i. The resulting logits ℓ_i^a score the finite vocabulary $\mathcal{W}$, and their difference is

$$\delta_i = \ell_i^S - \ell_i^I = \psi(x_{L,i}^I + \Delta x_{L,i}) - \psi(x_{L,i}^I). \tag{3}$$

Consistent with Equation 2, stale KV's key deviations after the reused span peak at intermediate layers and remain nonzero at the final layer (Figure 3, top left and middle).

When a score change becomes token divergence. The current token changes when another token's relative score increase exceeds its margin. At a shared generation position i, where both executions have the same prefix, write the logits as $\ell_i^S = \ell_i^I + \delta_i$. Greedy decoding selects $w_i^a \in \arg\max_{w \in \mathcal{W}} \ell_i^a(w)$, using the same fixed rule to break ties. Write $w_I = w_i^I$ and $w_S = w_i^S$. Each token $w \neq w_I$ is a competitor, whose margin $m_i(w) = \ell_i^I(w_I) - \ell_i^I(w) \geq 0$ is the current token's lead over w. The smallest margin $m_i^\star = \min_{w \neq w_I} m_i(w)$ is its lead over the runner-up.

Proposition 1 (Token change and preservation)**.** *Define*

$$\Gamma_i = \max_{w \neq w_I} \left\{ \underbrace{\delta_i(w) - \delta_i(w_I)}_{\textit{relative score change}} - \underbrace{m_i(w)}_{\textit{margin}} \right\}. \tag{4}$$

Positive Γ_i changes the token, negative Γ_i preserves it, and zero leaves the choice to the tie rule. Moreover, as in margin certificates (Hein & Andriushchenko, 2017; Tsuzuku et al., 2018),

$$2\|\delta_i\|_\infty < m_i^\star \quad \Longrightarrow \quad w_S = w_I, \tag{5}$$

and if this bound holds at every shared generation position, the two executions have identical responses, actions, and task states (proof in Appendix A.1).

Corollary 1 (Margin-limited divergence)**.** *Call a shared generation position i at risk when $2\|\delta_i\|_\infty \geq m_i^\star$, and let N_z count the at-risk positions that the two executions share. The first token change of a diverging pair occurs at an at-risk position i, and its new token trails the current one by $m_i(w_S) \leq 2\|\delta_i\|_\infty$. It follows that a pair without an at-risk position cannot diverge:*

$$P(D = 1 \mid z) \leq P(N_z \geq 1 \mid z). \tag{6}$$

Divergence can thus start only at near-tied decisions, where the logit change closes the lead.

Corollary 1 yields two predictions, which we test with native full-vocabulary logits on the 72 controlled tasks (Appendix C.2). First, the new token trails the current one by at most $2\|\delta_i\|_\infty$, so it should be a close competitor. It is the runner-up in 65 of 70 Qwen and 56 of 72 Ministral first changes and trails by a median of 0.75 and 0.38 logits (Appendix Table 9). Second, a token can change only when the logit change reaches the margin, so changes should become more frequent as $r_i = m_i^\star/(2\|\delta_i\|_\infty)$ decreases and vanish for $r_i > 1$. Within each model, 34.9% of Qwen and 34.8% of Ministral positions with $r_i < 0.25$ change tokens, against 0.5% and none for $0.5 \leq r_i \leq 1$ and none above one (Figure 3, top right).

Corollary 1 holds for any two caches, including current-KV loading and recomputation. Their numerical differences also flip near-tied tokens (Yuan et al., 2025), in 3 of 72 Qwen and 71 of 72 Ministral executions. On positions shared by all three executions, stale KV enlarges the mean logit deviation beyond this baseline by 3.05 in Qwen and 1.83 in Ministral, with task-bootstrap intervals above zero (Appendix Figure 7). In both models, stale KV reverses near-tied choices, more often at smaller margins and mostly toward the runner-up.

Why the direction of the difference matters. Raising both candidates' scores equally preserves the margin, whereas raising the competitor alone can overcome it. The layer and output maps project a KV difference onto each competitor's relative score (Appendix A.1). Hence, at equal KV norm, the token choice depends on the direction of the difference.

Two controls keep the norm of $\Delta\mathcal{K}_s$ and change its direction on 48 tool-schema pairs per model. Matched-norm noise uses seeded Gaussian noise, and the structure-preserving control flips the signs of seeded channels, preserving inner products between difference vectors (Appendix C.1). At the same norm, both controls leave larger key deviations at the final layer than stale KV (Figure 3, top left and middle). Stale keys alone, stale values alone, or stale KV in one third of the layers change tokens in nearly every task (Appendix Table 8). Even part of $\Delta\mathcal{K}_s$ is enough to change tokens, and its direction sets how far the change propagates.

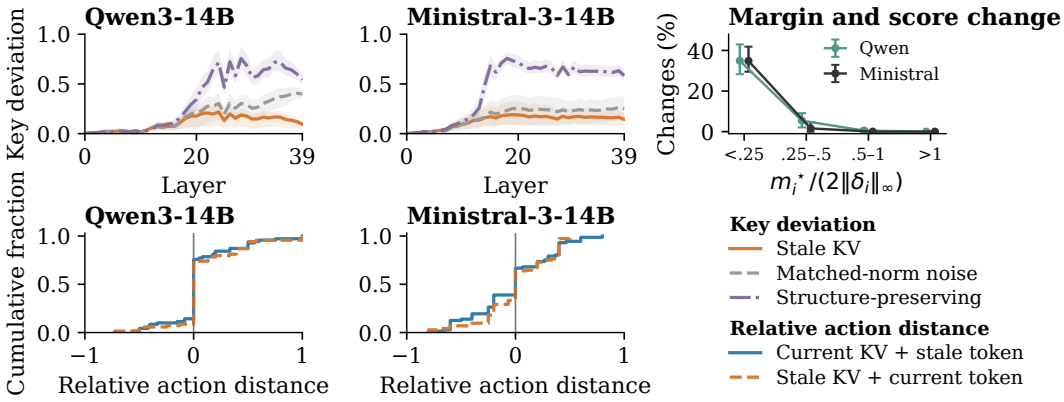

Figure 3: **Propagation, token sensitivity, and persistence.** Top left and middle: median relative prompt-key deviation per layer (Appendix Equation 34) on 48 tool-schema pairs per model, with interquartile bands. Top right: token-change rates by r_i, with 95% task-bootstrap intervals for Qwen (green) and Ministral (black). Bottom: cumulative fraction of valid continuations by relative action distance, the distance to the KV's execution minus that to the token's execution.

Why divergence persists after the first token. Every later decision depends on the generated history and on the span cache that stays loaded, which we call the retained KV. At turn t, let H_t^a be the history, R_t^a the response, and A_t^a its parsed action. The environment $\mathcal{E}_z$ updates the task state from W_{t-1}^a to W_t^a and returns observation O_t^a. The generator G builds the prompt from H_t^a and produces the response with the span cache $\mathcal{K}_t^a$ loaded at turn t. Writing $\oplus$ for appending to history, one turn is

$$\begin{aligned} R_t^a &= G(H_t^a; \mathcal{K}_t^a), \qquad A_t^a = \mathrm{parse}(R_t^a), \\ (W_t^a, O_t^a) &= \mathcal{E}_z(W_{t-1}^a, A_t^a), \quad H_{t+1}^a = H_t^a \oplus R_t^a \oplus O_t^a. \end{aligned} \tag{7}$$

A changed token can redirect the rest of the generation (Bigelow et al., 2025), as mistakes do under exposure bias (Bengio et al., 2015; Ross et al., 2011; Arora et al., 2022), and a changed action can alter subsequent observations. Matching the history, retained KV, and task state gives the same next response and environment transition under Equation 7. Restoring one input alone leaves the others in place.

When the two executions exchange their first divergent tokens, some continuations stay closer to the execution that supplied the token and others to the execution that supplied the KV (Figure 3, bottom). Restoring the current KV or history one to four turns after divergence, with the stale workspace kept, shortens Qwen's median action distance. No execution returns to the current path (Appendix Table 12). The retained KV shapes later behavior, and the divergent history and workspace keep the divergence after the current KV returns.

5 Why Can Tasks Still Succeed Despite Divergence?

To explain why tasks still succeed despite divergence, we follow divergent executions to their final states, intervene on their first changed token and on later feedback, and derive when divergence introduces failure. Divergent executions can succeed through unchanged actions, valid alternative paths, or corrective feedback, and the direction of the KV difference decides whether they do.

How different behavior can preserve success. Divergence begins where the model scores two tokens nearly equally (Corollary 1), so the stale execution starts from a continuation that the current

model itself scores almost as high. The current model should then also succeed from that continuation. Under current KV, continuations forced to the stale execution's first new token succeed in 138 of 140 Qwen and all 144 Ministral pairs (Appendix Table 10). Forcing the runner-up or the third-ranked token at the first divergence position of eight Qwen tasks succeeds in all eight tasks per condition and changes the actions in six (Appendix Table 11). Different responses also produce identical parsed actions in 16 of the 140 divergent Qwen pairs (Appendix Table 4). The task-success check reads only the final state, as in test-based agent benchmarks (Jimenez et al., 2024), so a direct edit and a script that writes the required content both pass. A near-tied decision thus opens several successful paths, and absorption holds when the retained stale KV stays on one.

Feedback can also restore progress after an unsuccessful action. A failed test or tool error enters the history by Equation 7 and lets the agent revise its next action (Shinn et al., 2023; Gou et al., 2024).

Absorption without state rejoining. The divergence turn is the first turn with different actions, or with different responses when actions never differ. *State rejoining* means that the two executions match in workspace and observations at or after the divergence turn, excluding observation runtime metadata (Appendix C.4). Among absorbed pairs, 90 of 140 in Qwen and 102 of 144 in Ministral never rejoin before termination (Figure 4). Still, 76 and 102 of these pairs end in the same final workspace (Appendix C.4). Most absorbed executions reach the same final state by another path.

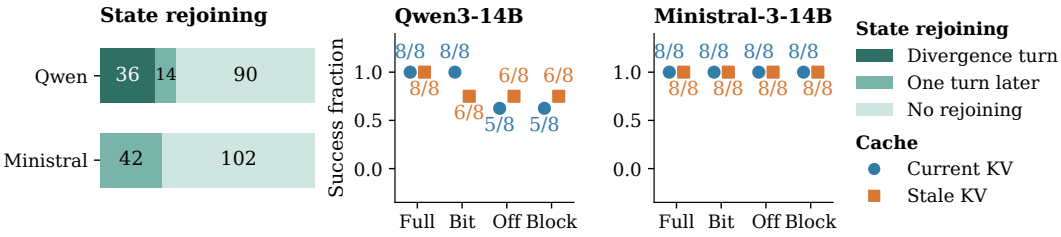

Figure 4: **Absorption and feedback dependence.** Left: rejoining counts among 140 Qwen and 144 Ministral absorbed pairs. Middle and right: successes of eight tasks resumed at the first failing test after divergence. Full gives test output, Bit pass or fail, Off nothing, and Block also stops reruns.

How feedback affects absorption. We continue eight tasks per model from the first failing test after divergence and vary access to test feedback (Appendix C.5). Without test feedback, Ministral still completes all eight tasks under both caches, whereas Qwen fails three with current KV and two with stale KV (Appendix Table 14). With only a success bit, stale KV fails two Qwen tasks that current KV completes. Under Bit, three successful replays leave changes outside the permitted file, two with stale KV and one with current KV. Feedback enters both histories through Equation 7, so removing it lowers current and stale success alike. Success can also hide an action violation.

How unabsorbed divergence causes introduced failure. Let $h(z) = P(Y_I = 1, Y_S = 0 \mid z)$ denote the probability of introduced failure. Let $\pi(z) = P(Y_I = 1, D = 1 \mid z)$ be the probability that a pair diverges and its current execution succeeds. For $\pi(z) > 0$, the absorption probability is

$$\alpha(z) = P(Y_S = 1 \mid Y_I = 1, D = 1, z). \tag{8}$$

Decomposition 2 (Introduced failure)**.** *Under the execution conditions of Section 3, introduced failure requires divergence, and for* $\pi(z) > 0$

$$h(z) = \underbrace{\pi(z)}_{\substack{\text{current success} \\ \text{and divergence}}} \underbrace{[1 - \alpha(z)]}_{\substack{\text{fraction not} \\ \text{absorbed}}}. \tag{9}$$

If $\pi(z) = 0$*, then* $h(z) = 0$*.*

Decomposition 2, proved in Appendix A.2, gives two routes to low introduced failure: unchanged behavior keeps $\pi(z)$ small, and absorption keeps $1 - \alpha(z)$ small. The controlled tasks follow the second route, since every divergence is absorbed (Appendix Table 4).

Table 1: Task outcomes of KV differences with matched norms and different directions. Key diff. is the median relative final-layer key deviation, Noise is matched-norm noise, and SP is the structure-preserving control. Key diff. and $\times 1$ use 48 tool-schema pairs per model, and $\times 0.5$ and $\times 2$ scale the difference on eight tool-schema tasks (Appendix Tables 7 and 8).

	Key diff.			Success						
				Stale KV			Noise			SP
Model	Stale KV	Noise	SP	$\times 0.5$	$\times 1$	$\times 2$	$\times 0.5$	$\times 1$	$\times 2$	$\times 1$
Qwen	0.095	0.399	0.544	8/8	48/48	0/8	6/8	0/48	0/8	0/48
Ministral	0.144	0.251	0.583	8/8	48/48	0/8	8/8	48/48	0/8	0/48

What limits absorption. By Corollary 1, a larger logit change can reverse decisions with larger margins, beyond the near-ties that stale KV reverses. At twice the norm of $\Delta\mathcal{K}_s$, both directions fail every task (Table 1). At the norm itself, stale KV completes every task, whereas the structure-preserving control fails every task in both models and matched-norm noise every Qwen task. Therefore, at equal norm, the direction of the KV difference decides the task outcome.

6 When Is Stale KV Cache Reuse Permissible?

To determine when a completed execution permits KV cache reuse, we combine task success with resource, workspace, and action requirements.

Expressing the complete execution requirements. For a trajectory τ, let $Y(\tau)$ denote task success, so $Y(\tau^a) = Y_a$. The nonnegative measures $C_z(\tau)$, $X_z(\tau)$, and $Q_z(\tau)$ count process cost, workspace side effects, and action violations. Examples include tool calls, extra file changes, and edits outside the permitted file, respectively. With nonnegative weights fixed in advance, define

$$\mathcal{L}_z(\tau) = \lambda_Y[1 - Y(\tau)] + \lambda_C C_z(\tau) + \lambda_X X_z(\tau) + \lambda_Q Q_z(\tau). \tag{10}$$

Connecting divergence and absorption to expected loss. The relative expected loss $\mathcal{R}(z) = \mathbb{E}_z[\mathcal{L}_z(\tau^S) - \mathcal{L}_z(\tau^I)]$ averages the stale-minus-current loss change over the pairs of setting z. Write $\bar{\Delta}_C(z) = \mathbb{E}_z[C_z(\tau^S) - C_z(\tau^I)]$, with $\bar{\Delta}_X(z)$ and $\bar{\Delta}_Q(z)$ defined analogously. Let $g(z) = P(Y_I = 0, Y_S = 1 \mid z)$ be the probability of gained success. For $\pi(z) > 0$, substitution of Equation 9 gives

$$\mathcal{R}(z) = \underbrace{\lambda_Y\{\pi(z)[1 - \alpha(z)] - g(z)\}}_{\text{task outcomes}} + \underbrace{\lambda_C\bar{\Delta}_C(z) + \lambda_X\bar{\Delta}_X(z) + \lambda_Q\bar{\Delta}_Q(z)}_{\text{execution consequences}}. \tag{11}$$

With full absorption ($\alpha(z) = 1$), only gained success and execution consequences remain.

From successful absorption to permissible execution. A small average loss can hide pairs that violate requirements, as unchanged accuracy hides changed answers (Dutta et al., 2024), so each pair must meet its own constraints. For a pair with $Y_I = Y_S = 1$, the task term cancels, leaving

$$\Delta\mathcal{L}_z = \lambda_C\Delta C_z + \lambda_X\Delta X_z + \lambda_Q\Delta Q_z. \tag{12}$$

Let $B_z = B_z(\tau^I, \tau^S)$ indicate whether this pair meets all execution requirements. These requirements cap each increase, $\Delta C_z \le \epsilon_C$, $\Delta X_z \le \epsilon_X$, and $\Delta Q_z \le \epsilon_Q$, with nonnegative budgets. B_z also requires parsable actions, permitted edits, and matching final workspaces.

Lemma 3 (Execution bound)**.** *For a pair with $Y_I = Y_S = 1$ and $B_z = 1$,*

$$\Delta\mathcal{L}_z \le \lambda_C\epsilon_C + \lambda_X\epsilon_X + \lambda_Q\epsilon_Q. \tag{13}$$

With zero budgets, $\Delta\mathcal{L}_z \le 0$ for every choice of nonnegative weights.

The criterion thus bounds the loss change for all weights without fixing them (Appendix A.3).

Table 2: Divergence, retention, and task outcomes by setting. Current-KV retention divides joint recomputation and current-KV successes by recomputation successes. $\hat{\alpha}$ divides absorbed pairs by pairs with $Y_I = 1$ and $D = 1$, and n_h and n_g count introduced failures and gained successes.

		Divergence		Success retention		Task outcomes	
Model	n	D_{tok}	D_{act}	Current KV	Stale KV ($\hat{\alpha}$)	n_h	n_g
Controlled tasks							
Qwen	144	140	124	144/144	140/140	0	0
Ministral	144	144	144	144/144	144/144	0	0
SWE-Skills-Bench							
Qwen	49	49	47	23/24	23/23	0	1
Ministral	49	49	49	40/41	42/44	2	2
SkillsBench							
Qwen	87	87	85	6/7	5/8	3	7
Ministral	87	87	87	37/46	24/41	17	10

Conditions for permissible reuse. For a valid pair with $Y_I = 1$, permissible KV cache reuse requires $Y_S = 1$ and $B_z = 1$. It is transparent when responses remain unchanged ($D = 0$) and absorption-backed when they diverge ($D = 1$). A stale failure or requirement violation prohibits reuse. Appendix Equation 29 gives the full rule and validity conditions. In Qwen, the criterion prohibits 28 successful pairs: 12 exceed a process budget and 16 leave workspace differences outside the task checks. All Ministral pairs are absorption-backed (Appendix Figure 11).

We also run 87 SkillsBench (Li et al., 2026) and 49 SWE-Skills-Bench (Han et al., 2026) tasks, with each task's skill text as the reused span and stale KV computed after another task's context (Appendix D). A task passes when at least half of its tests pass. Responses diverge on every public task, so $\hat{\alpha}$ in Table 2 estimates $\alpha(z)$ (Appendix Equation 31). It stays near one on SWE-Skills-Bench and drops to 5/8 and 24/41 on SkillsBench, where current-KV retention is 6/7 and 37/46. Stale content there lowers retention beyond numerical effects, and $\alpha(z)$ depends on the setting.

7 Discussion and Limitations

Three limitations of this work set our directions for future work.

Model scale. We use two 14B models under greedy decoding in one agent framework. At this scale, model capability limits which valid alternative paths the agent finds through planning, tool use, and feedback. Future work will test larger models and sampling to isolate stale-KV effects.

Multi-agent environments. Each of our executions involves a single agent. In multi-agent systems, agents share caches and exchange messages, so one agent's divergence enters the inputs of others. Future work will study how divergence spreads across agents and how they correct it or fail.

Criterion theory. Our criterion evaluates completed pairs against pre-specified requirements. It does not predict permissibility before execution or explain which semantic features of stale content set the KV difference's direction. Future work will address both and test them on new tasks.

8 Conclusion

To assess the reliability of cross-context KV cache reuse in LLM agent systems, we trace its effects on agent behavior and task success. We explain how KV differences change token choices and sustain behavioral divergence. The agent can still complete the task through a different sequence of actions or by correcting errors using feedback. Even when the task succeeds, the agent may exceed its resource budget or make unwanted changes to the workspace. We further derive a criterion that checks both task success and execution requirements to determine whether KV cache reuse is permissible. Experiments on controlled tasks and public benchmarks support our theory.

AI USE STATEMENT

In this work, we used generative AI tools for assisting with writing refinement and language polishing. We have not used generative AI tools for research ideation, research execution, retrieval and discovery of related work, drafting sections of the paper, generating synthetic datasets, or proving mathematical claims. Other AI-assisted tasks are not applicable to this work. Additionally, we reviewed all AI-assisted writing outputs and manually verified the accuracy, consistency, and correctness of the revised content. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 states the execution conditions behind our results, and Appendix A gives the definitions, assumptions, and complete proofs of Proposition 1, Corollary 1, Decomposition 2, and Lemma 3. Appendix B describes the construction of the 72 controlled tasks, their instructions and success checks, the agent and serving configuration of both models, and how current and stale KV are computed and loaded. Appendices B.2 and C.2 report replays that re-execute recorded requests and compare them with the original responses. Appendix C reports the per-model tables and figures behind the controlled results, and Appendix D describes the public-benchmark environments, execution limits, and pass threshold together with their results. The supplementary material contains anonymized code for the serving system, the KV connector, and the paired-execution runner, with instructions for reproducing the executions.

REFERENCES

Shubham Agarwal, Sai Sundaresan, Subrata Mitra, Debabrata Mahapatra, Archit Gupta, Rounak Sharma, Nirmal Joshua Kapu, Tong Yu, and Shiv Saini. Cache-Craft: Managing chunk-caches for efficient retrieval-augmented generation. *Proceedings of the ACM on Management of Data*, 3 (3):1–28, 2025.

Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Chi Kit Cheung. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 700–710, 2022.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Eric Bigelow, Ari Holtzman, Hidenori Tanaka, and Tomer Ullman. Forking paths in neural text generation. In *International Conference on Learning Representations*, 2025.

Valérie Castin, Pierre Ablin, and Gabriel Peyré. How smooth is attention? In *International Conference on Machine Learning*, pp. 5817–5840, 2024.

Abhinav Dutta, Sanjeev Krishnan, Nipun Kwatra, and Ramachandran Ramjee. Accuracy is not all you need. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Jiahai Feng, Stuart Russell, and Jacob Steinhardt. Monitoring latent world states in language models with propositional probes. In *International Conference on Learning Representations*, 2025.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt Cache: Modular attention reuse for low-latency inference. In *Proceedings of Machine Learning and Systems*, volume 6, 2024.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024.

Satchel Grant, Simon Jerome Han, Alexa R. Tartaglini, and Christopher Potts. Addressing divergent representations from causal interventions on neural networks. In *International Conference on Learning Representations*, 2026.

Tingxu Han, Yi Zhang, Wei Song, Chunrong Fang, Zhenyu Chen, Youcheng Sun, and Lijie Hu. SWE-Skills-Bench: Do agent skills actually help in real-world software engineering? *arXiv preprint arXiv:2603.15401*, 2026.

Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Junhao Hu, Wenrui Huang, Weidong Wang, Haoyi Wang, Tiancheng Hu, Qin Zhang, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. EPIC: Efficient position-independent caching for serving large language models. In *International Conference on Machine Learning*, pp. 24391–24402, 2025.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Hyunjik Kim, George Papamakarios, and Andriy Mnih. The Lipschitz constant of self-attention. In *International Conference on Machine Learning*, pp. 5562–5571, 2021.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 7057–7075, 2020.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation. In *International Conference on Learning Representations*, 2026.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, et al. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.

Yucheng Li, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir H. Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, and Lili Qiu. SCBench: A KV cache-centric analysis of long-context methods. In *International Conference on Learning Representations*, 2025.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Sichu Liang, Zhenglin Wang, Jiajia Chu, Pengfei Xia, Hui Zang, and Deyu Zhou. When KV cache reuse fails in multi-agent systems: Cross-candidate interaction is crucial for LLM judges. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7205–7230, 2026.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026a.

Yang Liu, Yunfei Gu, Liqiang Zhang, Chentao Wu, Guangtao Xue, Jie Li, Minyi Guo, Junhao Hu, and Jie Meng. CacheSlide: Unlocking cross position-aware KV cache reuse for accelerating LLM serving. In *Proceedings of the 24th USENIX Conference on File and Storage Technologies (FAST)*, pp. 83–99, 2026b.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. CacheGen: KV cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Dongyang Ma, Yan Wang, and Tian Lan. Block-Attention for efficient prefilling. In *International Conference on Learning Representations*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *International Conference on Learning Representations*, 2025.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 627–635, 2011.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, 2025.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Xinyu Yang, Tianqi Chen, and Beidi Chen. APE: Faster and longer context-augmented generation via adaptive parallel encoding. In *International Conference on Learning Representations*, 2025b.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, pp. 94–109, 2025.

Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, and Yiran Chen. KVCOMM: Online cross-context KV-cache communication for efficient LLM-based multi-agent systems. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. Understanding and mitigating numerical sources of nondeterminism in LLM inference. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations*, 2024.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. AgenTracer: Who is inducing failure in the LLM agentic systems? In *International Conference on Learning Representations*, 2026.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to AI transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A Theoretical Derivations

Definitions and domains. Cache differences compare arrays with the same token positions and tensor dimensions. For a token context H, let $\Phi(H)|_s$ collect the keys and values of span s across all layers and KV heads. The contexts H^{cur} and H^{src} contain identical span tokens after different preceding text. We set $\mathcal{K}_s^I = \Phi(H^{\mathrm{cur}})|_s$, and $\mathcal{K}_s^S$ is $\Phi(H^{\mathrm{src}})|_s$ with its keys rotated to the span's positions in H^{cur}. Both caches then occupy identical positions, so $\Delta\mathcal{K}_s$ compares them componentwise. Layers are indexed by $l = 0, \ldots, L-1$, and x_L denotes the final hidden representation. At layer l, let n_l be the number of KV heads and $d_{k,l}, d_{v,l}$ their key and value dimensions. Then $\mathcal{K}_s^a \in \prod_{l=0}^{L-1}(\mathbb{R}^{n_l \times |s| \times d_{k,l}} \times \mathbb{R}^{n_l \times |s| \times d_{v,l}})$, with differences taken componentwise. Equation 15 suppresses the head index and uses $d_k = d_{k,l}$ for the head being analyzed. Grouped-query attention uses the KV head assigned to each query head.

Execution assumptions and outcome indicators. We call the following setup matched deterministic execution. Each pair shares its initial prompt, history H_1, workspace W_0, tools, decoding configuration, and environment. Prompt construction, decoding, cache selection, parsing, and environment updates are deterministic given the instance and execution history. Both executions use the same stopping rules based on generated history and task state, including fixed token and turn limits. Each execution terminates after finitely many tokens and turns. Under these assumptions, a response difference identifies an effect of the loaded KV. For the controlled comparisons, we verify identical model-visible inputs through the first different response (Appendix B.2).

Trajectory indicators compare the complete executions. The trajectory τ^a records responses, actions, observations, task states, and resource use. Let T_a be the final turn and χ_z the binary checker that tests whether the final task state satisfies the requirements. For $t > T_a$, set $R_t^a = A_t^a = \perp$ and compare turns up to $\max(T_I, T_S)$. The marker $\perp$ differs from every actual response and parsed action. The indicators in Section 3 are

$$D_{\mathrm{tok}} = \mathbf{1}[\exists t : R_t^I \neq R_t^S], \qquad D_{\mathrm{act}} = \mathbf{1}[\exists t : A_t^I \neq A_t^S], \qquad Y_a = \chi_z(W_{T_a}^a). \tag{14}$$

A fixed parser gives $D_{\mathrm{act}} = 1 \Rightarrow D_{\mathrm{tok}} = 1$, while equal responses can still incur different resource use. Equation 10 accounts for that resource use through the measures defined in Section 6.

A.1 Divergence and Persistence

Attention differences. The attention decomposition compares the same query position and visible token positions under the two caches. For one head at layer l, let $q_{l,i}, k_{l,j}, v_{l,j}$ denote the query, key, and value vectors, with key dimension d_k. Causal attention computes

$$\beta_{l,ij} = \mathrm{softmax}_{j\le i}\Big(q_{l,i}^\top k_{l,j}/\sqrt{d_k}\Big), \qquad o_{l,i} = \sum_{j\le i} \beta_{l,ij} v_{l,j}. \tag{15}$$

Queries and keys include positional transformations. Here $\Delta v_{l,j}$ ranges over all attended positions, including value changes that earlier layers carry past the reused span. Suppressing l, i and adding and subtracting $\sum_j \beta_j^S v_j^I$ gives

$$\begin{aligned} o^S - o^I &= \sum_j \beta_j^S v_j^S - \sum_j \beta_j^I v_j^I \\ &= \sum_j (\beta_j^S - \beta_j^I) v_j^I + \sum_j \beta_j^S (v_j^S - v_j^I). \end{aligned} \tag{16}$$

This exact identity gives Equation 1 for KV differences of any size.

Local propagation and output scores. The expansion below splits each layer's output difference into linear terms and an exact remainder. Let F_l be layer l in exact arithmetic, a differentiable map, with x_l stacking hidden representations after the reused span. The argument $\mathcal{K}_l$ is the layer-l component of the span cache $\mathcal{K}_s$. KV preceding the span is fixed and folded into F_l. Write $x_{l+1}^a = F_l(x_l^a, \mathcal{K}_l^a) + \varepsilon_l^a$, where ε_l^a is the numerical residual and is zero in exact arithmetic. After vectorization, write $x_l \in \mathbb{R}^{d_l}$ and $\mathcal{K}_l \in \mathbb{R}^{d_l^{\mathcal{K}}}$. Thus $J_l^x \in \mathbb{R}^{d_{l+1}\times d_l}$ and $J_l^{\mathcal{K}} \in \mathbb{R}^{d_{l+1}\times d_l^{\mathcal{K}}}$. All other inputs and the shared token sequence are fixed. Assume F_l is continuously differentiable on a neighborhood of the line segment between the current and stale arguments. At the current arguments define $J_l^x = \partial_x F_l(x_l^I, \mathcal{K}_l^I)$ and $J_l^{\mathcal{K}} = \partial_{\mathcal{K}} F_l(x_l^I, \mathcal{K}_l^I)$. For $u_l = (\Delta x_l, \Delta \mathcal{K}_l)$ and $\Delta\varepsilon_l = \varepsilon_l^S - \varepsilon_l^I$, the full remainder is

$$\eta_l = \int_0^1 \big[\partial F_l\big((x_l^I, \mathcal{K}_l^I) + r u_l\big) - \partial F_l(x_l^I, \mathcal{K}_l^I)\big] u_l \, dr + \Delta\varepsilon_l. \tag{17}$$

The fundamental theorem of calculus applied along the segment yields Equation 2 exactly with this remainder, for differences of any size. If ∂F_l is Lipschitz on the segment with constant M_l, its integrand has norm at most $M_l r\|u_l\|^2$. Integration yields $\|\eta_l - \Delta\varepsilon_l\| \le M_l\|u_l\|^2/2$ in compatible vector and induced operator norms.

Repeated substitution of Equation 2 collects contributions from every layer. Identical token embeddings give $\Delta x_0 = 0$, so all differences enter through the layer KV terms and numerical residuals. For $l' > l$, define $\mathcal{T}_{l',l} = J_{l'-1}^x \cdots J_l^x$, with $\mathcal{T}_{l,l}$ the identity map. For each depth $n = 1, \ldots, L$,

$$\Delta x_n = \sum_{l=0}^{n-1} \mathcal{T}_{n,l+1}\big(J_l^{\mathcal{K}} \Delta\mathcal{K}_l + \eta_l\big). \tag{18}$$

For $n = 1$, this is Equation 2 at $l = 0$. If it holds at depth $n < L$, multiplying it by J_n^x and adding $J_n^{\mathcal{K}}\Delta\mathcal{K}_n + \eta_n$ gives depth $n + 1$. Induction therefore gives the expansion at the final depth $n = L$.

The output map ψ in Equation 3 applies final normalization and the vocabulary projection as executed, so the token criterion of Proposition 1 compares the actual logits.

The output map in exact arithmetic, ψ_0, makes the dependence on KV direction explicit. It is continuously differentiable along the compared representations. Write $\varepsilon_{\psi,i}^a = \psi(x_{L,i}^a) - \psi_0(x_{L,i}^a)$ and $\Delta\varepsilon_{\psi,i} = \varepsilon_{\psi,i}^S - \varepsilon_{\psi,i}^I$.

Projecting the output difference onto two token coordinates gives their relative score change. Let Π_i select position i from the stacked vector x_L, and let $c_w = \mathbf{e}_w - \mathbf{e}_{w_I}$ subtract the current token's coordinate from competitor w. Define $\tilde{c}_{i,w}^\top = c_w^\top \partial\psi_0(x_{L,i}^I)\Pi_i$. Expanding ψ_0 and substituting Equation 18 at $n = L$ gives

$$\delta_i(w) - \delta_i(w_I) = \sum_{l=0}^{L-1} \tilde{c}_{i,w}^\top \mathcal{T}_{L,l+1} J_l^{\mathcal{K}} \Delta\mathcal{K}_l + \xi_{i,w}. \tag{19}$$

Here $\xi_{i,w} = \tilde{c}_{i,w}^\top \sum_{l=0}^{L-1} \mathcal{T}_{L,l+1}\eta_l + c_w^\top \zeta_i$, where ζ_i retains the output remainder. Explicitly, $\zeta_i = \int_0^1 [\partial\psi_0(x_{L,i}^I + r\Delta x_{L,i}) - \partial\psi_0(x_{L,i}^I)]\Delta x_{L,i}\, dr + \Delta\varepsilon_{\psi,i}$. Each term in the sum projects the layer-l KV difference onto the relative score of competitor w.

A linear projection of a vector depends on its direction at equal norm. For every $\gamma > 0$, the vectors $(\gamma, 0)$ and $(0, \gamma)$ share the Euclidean norm γ, and the row $(1, 0)$ maps them to γ and zero. A two-token example turns this into different greedy choices. Take a vocabulary $\{w_I, w\}$ with margin $m_i(w) = \gamma/2$, zero remainders, and a relative-score row equal to $(1, 0)$. The perturbation $(\gamma, 0)$ gives $\Gamma_i = \gamma - \gamma/2 > 0$ and changes the token, while $(0, \gamma)$ gives $\Gamma_i = -\gamma/2 < 0$ and preserves it. Thus, equal-norm KV differences can yield different relative score changes and token choices.

Equations 18 and 19 cover both the smooth model and finite-precision executions. In the smooth model, $\varepsilon_l^a = \varepsilon_{\psi,i}^a = 0$ and $\psi_0 = \psi$. In a quantized execution, their differences remain in η_l and ζ_i, so the identities differentiate only the exact-arithmetic maps F_l and ψ_0.

Proof of Proposition 1. Proposition 1 applies to finite logits over a finite vocabulary containing at least two tokens. Greedy decoding uses a fixed tie rule. For the current choice w_I, define each competitor's margin as

$$m_i(w) = \ell_i^I(w_I) - \ell_i^I(w) \geq 0, \qquad w \neq w_I. \tag{20}$$

Substitution of $\ell_i^S = \ell_i^I + \delta_i$ gives

$$\begin{aligned} \ell_i^S(w) - \ell_i^S(w_I) &= \ell_i^I(w) - \ell_i^I(w_I) + \delta_i(w) - \delta_i(w_I) \\ &= \delta_i(w) - \delta_i(w_I) - m_i(w). \end{aligned} \tag{21}$$

Taking the maximum over $w \neq w_I$ yields Γ_i in Equation 4. If $\Gamma_i < 0$, w_I remains the unique maximizer. If $\Gamma_i > 0$, a competitor outranks w_I, so the selected token changes. At $\Gamma_i = 0$, w_I shares the maximum and the fixed tie rule determines the choice.

A uniform bound on score changes is sufficient for preservation. For every competitor, $\delta_i(w) - \delta_i(w_I) \leq |\delta_i(w)| + |\delta_i(w_I)| \leq 2\|\delta_i\|_\infty$. With $m_i^\star = \min_{w \neq w_I} m_i(w)$,

$$\Gamma_i \leq 2\|\delta_i\|_\infty - m_i^\star, \qquad 2\|\delta_i\|_\infty < m_i^\star \implies w_S = w_I. \tag{22}$$

The preservation bound in Equation 22 extends to complete behavior under matched deterministic execution. Assume identical initial inputs and task states, fixed parsing and decoding, and identical deterministic environment and stopping rules. Each pair fixes its source context and any cache-selection state. A deterministic cache policy in each condition loads $\mathcal{K}_t^a$ from the pair's record (Appendix A.2) and the execution history. Assume each execution terminates after finitely many tokens and turns, with termination controlled by the same history and task-state rules. At each shared generation position before termination, suppose the score bound holds for the actual two KV conditions. The bound gives identical next tokens, including any end-of-response token. Token induction preserves the whole response and its termination. The parser then returns identical actions, and the environment returns identical task states and observations. The next request thus has the same visible history. Repeating this argument across requests gives $D = D_{\text{act}} = 0$ and $Y_I = Y_S$, which completes Proposition 1.

Proof of Corollary 1. Corollary 1 follows from the contrapositive of Proposition 1. If $D = 1$, let i_0 be the first generation position at which the two executions select different tokens. Its prefix is shared, so $\Gamma_{i_0} \geq 0$, and Equation 22 gives $m_{i_0}^\star \leq 2\|\delta_{i_0}\|_\infty$. Position i_0 is therefore at risk, so $\{D = 1\} \subseteq \{N_z \geq 1\}$. The stale execution selects w_S at i_0, so $\ell_{i_0}^I(w_S) + \delta_{i_0}(w_S) \geq \ell_{i_0}^I(w_I) + \delta_{i_0}(w_I)$, which gives $m_{i_0}(w_S) \leq \delta_{i_0}(w_S) - \delta_{i_0}(w_I) \leq 2\|\delta_{i_0}\|_\infty$. Taking probabilities under the pair distribution μ_z of Appendix A.2 gives Equation 6. If $2\|\delta_i\|_\infty \leq \theta_z$ at every shared position, every at-risk position has $m_i^\star \leq \theta_z$, so divergence also requires a smallest margin of at most θ_z. The argument uses only the two logit vectors, so it applies to any two cache conditions.

Agent interaction with the environment. Generated history and retained KV affect later decisions through distinct inputs. Let H_t^a be visible history before turn t, and W_t^a the task state after that turn. The model generates through G, and the deterministic environment $\mathcal{E}_z$ returns an observation O_t^a. Starting from the shared initial state W_0, these maps give Equation 7. Here $\oplus$ appends

information, and $\mathcal{K}_t^a$ contains the span KV loaded at turn t. The single-request analysis suppresses the turn index, writing $\mathcal{K}_s^a$ for the full span cache and $\mathcal{K}_l^a$ for its layer component. The current KV depends on the current request, while stale KV cache reuse loads KV computed under the source context.

A.2 Absorption and Feedback

Matched-pair population. The probabilities and expectations in Sections 4 to 6 describe a population of pairs. Let the record ω fix a task instance, its initial history and state, its source context, and any state used to select caches. A specified distribution μ_z samples these records as $\omega \sim \mu_z$. Decoding is deterministic, so μ_z carries all randomness of the comparison. Both executions use the same record and differ only in the loaded cache, giving $\tau^I(\omega)$ and $\tau^S(\omega)$. Measurability is assumed for every outcome, loss, divergence, feedback-event, and requirement map, including B_z and any instance-dependent budget. The loss components C_z, X_z, and Q_z are integrable under both executions. Reported fractions assign equal weight to each pair. The theoretical identities apply to any such specified measure.

Accepted executions and absorption. The success check partitions complete executions by their terminal task state. Let $W_T(\tau)$ denote the final state of trajectory τ, and define the accepted set $\mathcal{V}_z = \{\tau : \chi_z(W_T(\tau)) = 1\}$. For one pair, the absorption indicator is $\mathcal{A} = \mathbf{1}[D = 1, Y_I = 1, Y_S = 1]$, and substituting $Y_a = \chi_z(W_{T_q}^a)$ gives $\mathcal{A} = \mathbf{1}[D = 1, \tau^I \in \mathcal{V}_z, \tau^S \in \mathcal{V}_z]$. Membership in $\mathcal{V}_z$ permits different responses, actions, and final states that satisfy the same task requirements. A divergent execution reaches $\mathcal{V}_z$ through unchanged actions, through changed actions that end in an accepted final state, or through corrective feedback that enters H_{t+1}^S via Equation 7 and guides a later successful action. The same trajectory can combine the last two routes.

Proof of Decomposition 2. Introduced failure requires divergence under matched deterministic execution. If $D = 0$, responses and their termination coincide. Induction over turns with the fixed parser and environment gives identical task states and outcomes, so $Y_I \neq Y_S$ implies $D = 1$. Under the pair distribution μ_z, let $\pi(z) = P(Y_I = 1, D = 1 \mid z)$. When $\pi(z) > 0$, Equation 8 defines the absorption probability $\alpha(z)$. The conditional-probability rule gives

$$\begin{aligned} h(z) &= P(Y_I = 1, Y_S = 0, D = 1 \mid z) \\ &= P(Y_I = 1, D = 1 \mid z) P(Y_S = 0 \mid Y_I = 1, D = 1, z) \\ &= \pi(z)[1 - \alpha(z)]. \end{aligned} \tag{23}$$

When $\pi(z) = 0$, the introduced-failure event has probability zero and $\alpha(z)$ is undefined. For positive current-success probability, $\pi(z) = P(Y_I = 1 \mid z)P(D = 1 \mid Y_I = 1, z)$. Consequently, at a fixed current-success rate, the divergence rate $P(D = 1 \mid Y_I = 1, z)$ and the unabsorbed fraction $1-\alpha(z)$ set $h(z)$.

Feedback interventions. Feedback dependence concerns outcomes under interventions on a specified information source. Let $\bar{z}$ fix the model, the type of reusable text, and the environment. Write $z(\varphi) = (\bar{z}, \varphi)$ for this setting under feedback condition φ. All feedback conditions use the same records and distribution $\mu_{\bar{z}}$, so $\tau^a(\omega; \varphi)$, $Y_a(\omega; \varphi)$, and $D(\omega; \varphi)$ change only through the feedback condition φ. Let φ, φ' be two feedback conditions, and $Y_a^{(\kappa)}(\varphi)$ the success of selected pair κ under cache condition a. Use a common set of $N \geq 1$ selected pairs with valid executions in all compared conditions. With $\bar{Y}_a(\varphi) = N^{-1} \sum_{\kappa=1}^N Y_a^{(\kappa)}(\varphi)$, define

$$\Delta_{\mathrm{FB}}(\varphi, \varphi') = [\bar{Y}_S(\varphi) - \bar{Y}_S(\varphi')] - [\bar{Y}_I(\varphi) - \bar{Y}_I(\varphi')]. \tag{24}$$

A positive value means the signed φ-versus-φ' success contrast is larger with stale KV. Zero means equal average changes under both caches, as when both success rates fall by the same amount. For an individual pair κ, $Y_S^{(\kappa)}(\varphi) = 1$ and $Y_S^{(\kappa)}(\varphi') = 0$ identify dependence on that intervention. To isolate loss of absorption, additionally require $Y_I^{(\kappa)}(\varphi) = Y_I^{(\kappa)}(\varphi') = 1$ and $D^{(\kappa)}(\varphi) = D^{(\kappa)}(\varphi') = 1$.

Feedback labels. Let T_{act} be the first turn with different normalized actions. The counts f_I and f_S record detected corrective feedback events from T_{act} onward. Let $U = 1$ indicate that both counts

and T_{act} are available, and define $\nu = \mathbf{1}[f_S > f_I]$ on these pairs. For $\pi(z) > 0$, partitioning the divergent pairs by record availability and ν gives

$$\begin{aligned}\alpha(z) = P(Y_S = 1, U = 0 \mid Y_I = 1, D = 1, z) \\ + P(Y_S = 1, U = 1, \nu = 0 \mid Y_I = 1, D = 1, z) \\ + P(Y_S = 1, U = 1, \nu = 1 \mid Y_I = 1, D = 1, z).\end{aligned} \tag{25}$$

A.3 Execution Risk and Reuse Requirements

Task outcomes and full trajectory loss. Expected loss accounts for both introduced failures and gained successes. Assume finite nonnegative weights and integrable $C_z(\tau^a), X_z(\tau^a), Q_z(\tau^a)$ for $a \in \{I, S\}$. Binary outcomes satisfy the per-pair identity

$$Y_I - Y_S = \mathbf{1}[Y_I = 1, Y_S = 0] - \mathbf{1}[Y_I = 0, Y_S = 1]. \tag{26}$$

Taking expectations gives $\mathbb{E}_z[Y_I - Y_S] = h(z) - g(z)$. Define the relative expected loss by

$$\mathcal{R}(z) = \mathbb{E}_z[\Delta\mathcal{L}_z], \qquad \Delta\mathcal{L}_z = \mathcal{L}_z(\tau^S) - \mathcal{L}_z(\tau^I). \tag{27}$$

Substituting Equation 10 and using linearity of expectation for the finite sum gives

$$\mathcal{R}(z) = \lambda_Y[h(z) - g(z)] + \lambda_C\bar{\Delta}_C(z) + \lambda_X\bar{\Delta}_X(z) + \lambda_Q\bar{\Delta}_Q(z). \tag{28}$$

For $\pi(z) > 0$, substituting $h(z) = \pi(z)[1 - \alpha(z)]$ gives Equation 11. For $\pi(z) = 0$, Equation 28 applies directly with $h(z) = 0$.

Proof of Lemma 3. Lemma 3 applies to pairs that already succeed under both caches. For such a pair, subtracting the two losses cancels the task term and gives Equation 12. Here $\Delta C_z = C_z(\tau^S) - C_z(\tau^I)$, with corresponding definitions for $\Delta X_z, \Delta Q_z$. Let $B_z = 1$ require every component increase to stay within its nonnegative budget, together with any per-execution and pairwise requirements. Multiplying each component inequality by its nonnegative weight and summing proves Lemma 3. Setting every budget to zero gives $\Delta\mathcal{L}_z \le 0$. If success and $B_z = 1$ hold almost surely, taking expectations bounds $\mathcal{R}(z)$ by the weighted budget sum. For instance-dependent budgets, use the expectation of that sum and assume it is integrable.

Paired reuse assessment. The assessment rule evaluates preservation of current successes under specified requirements. Let $\mathsf{e}_{\text{valid}} = 1$ indicate complete records, verified cache loading, and no response cut off at the output limit. The rule uses observed task outcomes and the requirements in B_z:

$$\text{Decision}(\tau^I, \tau^S) = \begin{cases} \text{unassessed}, & \mathsf{e}_{\text{valid}} = 0 \text{ or } Y_I = 0, \\ \text{prohibited}, & \mathsf{e}_{\text{valid}} = 1,\ Y_I = 1,\ (Y_S = 0 \text{ or } B_z = 0), \\ \text{transparent}, & \mathsf{e}_{\text{valid}} = 1,\ Y_I = Y_S = 1,\ B_z = 1,\ D = 0, \\ \text{absorption-backed}, & \mathsf{e}_{\text{valid}} = 1,\ Y_I = Y_S = 1,\ B_z = 1,\ D = 1. \end{cases} \tag{29}$$

The cases are disjoint and exhaustive. Transparent KV cache reuse requires identical responses, since $D = D_{\text{tok}}$. A pair with $Y_I = 0$ has no successful current execution whose preservation can be assessed.

Setting-level retention. The per-pair assessment in Equation 29 evaluates completed executions. Retention summarizes task outcomes over the pairs of a setting. For $P(Y_I = 1 \mid z) > 0$, define the retention probability

$$\sigma(z) = P(Y_S = 1 \mid Y_I = 1, z). \tag{30}$$

For $\pi(z) > 0$, decomposing over divergence gives

$$\sigma(z) = P(D = 0 \mid Y_I = 1, z) + P(D = 1 \mid Y_I = 1, z) \cdot \alpha(z). \tag{31}$$

Under matched deterministic execution, $D = 0$ implies $Y_S = Y_I$ (Appendix A.1), giving the first term. The second term uses $\alpha(z) = P(Y_S = 1 \mid Y_I = 1, D = 1, z)$ from Equation 8. Rearranging,

$$1 - \sigma(z) = P(D = 1 \mid Y_I = 1, z) \cdot [1 - \alpha(z)] = \frac{h(z)}{P(Y_I = 1 \mid z)}. \tag{32}$$

For $\pi(z) = 0$, no current success diverges, so $\sigma(z) = 1$ and $h(z) = 0$. In both cases, the complement of retention equals the conditional introduced-failure rate. Full absorption ($\alpha(z) = 1$) gives $\sigma(z) = 1$ regardless of divergence rate.

Given N pairs from setting z, let N_I count those with $Y_I = 1$ and N_{IS} those with $Y_I = Y_S = 1$. The retention estimate is $\hat{\sigma}(z) = N_{IS}/N_I$ for $N_I > 0$. Table 17 reports N_I and N_{IS} as the tasks passing under current KV and under both current and stale KV. Under the full criterion, N_{perm} counts the pairs assessed as transparent or absorption-backed, and N_{perm}/N_I is the corresponding retention (Table 15).

B Tasks and Experimental Setup

B.1 Task Construction and Coverage

Each controlled task asks the agent to set one field in a small workspace by following the rules in a context object. A context object is one skill document, tool schema, or policy instruction, and we append its rule block under the heading *Operational Contract*. The requested work is a bounded file edit.

Each type of reusable text contains four context objects and six selected tasks per object, as shown in Table 3. The three types contribute 72 tasks in total. Each model runs every task under two seeds, giving 144 pairs, with 48 pairs per type. A pair is one task under one seed, executed with current and stale KV. Both seeds give the same divergence and reuse assessment on every task. Mechanism analyses use these executions or selected continuations from them. Table 4 gives the divergence and absorption counts of these pairs.

Table 3: Controlled tasks used in the mechanism comparisons. Counts of pairs are per model.

Reusable text	Context objects	Tasks	Pairs
Skill documents	Word, spreadsheet, presentation, and tool-building instructions	24	48
Tool schemas	Terminal with planning, editor with planning, terminal with editor, and edit with skill invocation	24	48
Policy instructions	Default, planning, security, and documentation agent policies	24	48
Total	12 context objects	72	144

Table 4: Divergence and absorption in the controlled tasks. Absorption and no-rejoining entries give counts over the divergent pairs. State rejoining compares observations and workspace state (Section 5).

Model	n	D_{tok}	D_{act}	Absorbed	No rejoining
Qwen3-14B	144	140	124	140/140	90/140
Ministral-3-14B	144	144	144	144/144	102/144

The tasks cover eight fields. These specify an artifact format, retry limit, timeout, severity ordering, validation mode, record prefix, provenance flag, or final status. Their target files are JSON, Python, TOML, plain text, configuration text, or Markdown. Each task bundle contains an initial workspace, an executable checker, and a reference edit.

Task selection retains reproducibly solvable candidates. A Qwen3-8B pilot agent executes each candidate twice under recomputation. A candidate passes when both runs pass the checker, finish on their own, edit only the permitted file, and leave complete records. Each run must also contain at least one corrective-feedback event and keep every prompt within 32,768 tokens. For each context object, the passing candidates are ordered by bundle hash, and the first six are retained.

B.2 Agent Execution and Success Checks

Qwen3-14B-FP8 and Ministral-3-14B-FP8 each have 40 layers, with their own attention layouts, tokenizers, and rotary position settings. Both use greedy decoding with thinking disabled within the same OpenHands and vLLM workflow. Each execution allows at most 20 turns in a 40,960-token context, with at most 8,192 output tokens per response for Qwen and 4,096 for Ministral. The comparisons match prompt tokens, tool definitions, initial workspaces, and decoding configuration. Re-executing 60 divergent pairs per model from the divergent request reproduces each cache's original response exactly.

The captured requests expose five executable tool interfaces. Table 5 summarizes their arguments and roles in the task workflow. Tool-schema context objects add reusable description text to the terminal tool.

Table 5: Executable tools in the captured requests. Arguments shown are the fields relevant to the task workflow.

Tool	Representative arguments	Role and returned information
`terminal`	`command`, `timeout`	Runs shell commands and tests, returning output and exit status.
`file_editor`	`command`, `path`, `old_str`, `new_str`	Views and edits files, returning file contents or editing feedback.
`task_tracker`	`command`, `task_list`	Views and updates a structured task list.
`think`	`thought`	Records intermediate reasoning without obtaining new workspace information.
`finish`	`message`	Ends the agent execution with a final response.

Each execution starts from a fresh copy of the same task workspace. The agent chooses its tool calls and receives the resulting observations until it finishes or reaches the turn limit. The recorded responses, actions, observations, and file changes support the mechanism analyses.

The final workspace determines task success. Each checker tests the requested field of the target file. Changes outside the permitted file are recorded separately from checker success and violate the action requirements in B_z (Section 6). Different actions and workspaces can therefore pass the same task check.

Generated responses define token divergence, and normalized action sequences define action divergence. The divergence turn is the first turn with different actions, or with different responses when the actions never differ. State rejoining is classified as occurring in that turn, one turn later, or not before termination. Final workspace comparisons record differences outside the task checks.

Matched model-visible inputs. All controlled comparisons begin from identical workspace contents and logical workspace paths. We compare the recorded model-visible messages, tool schemas, prompt tokens, and generation settings at each request through the first response difference. These inputs match in every current-versus-stale and recomputation-versus-current pair, in both the original runs and the native-logit runs. The native-logit analysis additionally requires identical generated prefixes at each compared position. Every original injected request records a completed, verified load at all 40 layers. Every original response parses, and no original execution violates its final changed-file contract.

Across the two seeds, every Qwen task repeats its greedy trajectory under all three conditions, and every Ministral task does so except one under recomputation and one under current KV. Divergence labels and execution assessments agree across seeds for every task. We use tasks as the resampling unit for the native-logit comparisons.

B.3 Task Instructions and Examples

Six task instances illustrate two fields from each type of reusable text. The boxes show the requested edit, the active requirement, the initial file, and the reference completion.

Skill documents. The skill examples request a numerical timeout and a literal severity order. Both values come from the active skill's Operational Contract.

Timeout from a presentation skill

Requested edit. Replace only the TODO placeholder in `runtime.toml` with the literal value registered for `timeout_seconds`.

Active requirement. `timeout_seconds = 60`

Initial file: `runtime.toml`

```
timeout_seconds = 0 # TODO
```

Reference completion

```
timeout_seconds = 60
```

Severity ordering from a Word-document skill

Requested edit. Replace only the TODO placeholder in `severity.txt` with the literal value registered for `severity_order`.

Active requirement. `severity_order = "critical-medium-high-low"`

Initial file: `severity.txt`

```
TODO
```

Reference completion

```
critical-medium-high-low
```

Tool schemas. The tool-schema examples request a retry limit and a final status.

Retry limit from a tool-schema context

Requested edit. Replace only the TODO placeholder in `retry_policy.py` with the literal value registered for `retry_limit`.

Active requirement. `retry_limit = 5`

Initial file: `retry_policy.py`

```
RETRY_LIMIT = -1 # TODO
```

Reference completion

```
RETRY_LIMIT = 5
```

Final status from a tool-schema context

Requested edit. Replace only the TODO placeholder in `STATUS.md` with the literal value registered for `final_status`.

Active requirement. `final_status = "verified"`

Initial file: `STATUS.md`

```
# Delivery Status
TODO
```

Reference completion

```
# Delivery Status
verified
```

Policy instructions. The policy examples request an artifact format and a provenance flag.

Artifact format from an agent policy

Requested edit. Replace only the TODO placeholder in `artifact.json` with the literal value registered for `artifact_format`.

Active requirement. `artifact_format = "markdown"`

Initial file: `artifact.json`

```
{"artifact_format": "TODO"}
```

Reference completion

```
{"artifact_format": "markdown"}
```

Provenance flag from a documentation-agent policy

Requested edit. Replace `null` in `provenance.json` with the literal value registered for `provenance_required`.

Active requirement. `provenance_required = true`

Initial file: `provenance.json`

```
{"provenance_required": null}
```

Reference completion

```
{"provenance_required": true}
```

B.4 Prompt Construction and KV Comparisons

The current request combines the task instruction with its active context object. In the native agent prompt, skill documents appear in the user message, policy instructions in the system message, and tool-schema material in the terminal tool description. Table 6 compares the active and preceding source requirements for three of the task instances.

Table 6: Source and current contexts for three task instances. The preceding source value belongs to the context before the unchanged reused text.

Material and task	Initial workspace file	Active requirement	Source context
Presentation skill, timeout	`runtime.toml` `timeout_seconds = 0 # TODO`	`timeout_seconds` equals `60`	Prior-session context contains `75`
Tool schema, retry limit	`retry_policy.py` `RETRY_LIMIT = -1 # TODO`	`retry_limit` equals `5`	Cross-agent context contains `3`
Agent policy, artifact format	`artifact.json` `{"artifact_format": "TODO"}`	`artifact_format` equals `"markdown"`	Nested context contains `"yaml"`

Across all 72 captured tasks in both models, the reused text and its token sequence match exactly between the source and current requests. The source request adds preceding material from a prior session, another agent context, or nested context. This material gives other values for some fields of the same rules, including the task's field in 55 of the 72 tasks. We search every edit and command of the 110 executions per model on these 55 tasks for the source value of the task's field. No stale execution among these 110 writes the source value. Across all 144 stale executions per model, the current value appears in the edits of 138 Qwen and 144 Ministral executions. Each of the three sources supplies 24 tasks per model, two per context object.

The agent-policy example in Table 6 shows how the source context differs. Its source request places a contract containing `artifact_format = "yaml"` before the reused text. The reused text itself specifies `artifact_format = "markdown"` in both source and current requests. During the comparison, both executions receive the current visible request and the same initial workspace.

The source span occupies later positions than the current span. The stale cache is moved to the current positions by rotating its keys with the model's own positional encoding, while values stay unchanged. Both caches are then loaded through the same procedure, so only the loaded KV for the selected span changes, as defined in Section 3.

Each of the 144 pairs also has a standard recomputation, giving 432 executions per model. Relative to recomputation, loading the current KV changes 6 of 144 Qwen responses and 6 parsed actions. It changes 142 of 144 Ministral responses and 130 parsed actions, while all 432 executions per model succeed. Current and stale KV share the loading procedure, so their comparison isolates cache content.

C Supporting Evidence

C.1 Propagation and Direction

Direction controls. The propagation and direction comparisons use 48 tool-schema pairs per model. Matched-norm noise $\Delta\mathcal{K}^N$ draws seeded Gaussian differences whose combined key and value Frobenius norm matches the stale-minus-current difference in each layer. With layerwise components ΔK_l^N and ΔV_l^N, the constraint is

$$\|\Delta K_l^N\|_F^2 + \|\Delta V_l^N\|_F^2 = \|K_l^S - K_l^I\|_F^2 + \|V_l^S - V_l^I\|_F^2. \tag{33}$$

The structure-preserving control $\Delta\mathcal{K}^{N_{\mathrm{sp}}}$ flips the signs of seeded channels along the head dimension, separately for each layer's keys and values. This orthogonal transformation is shared across tokens and heads and preserves the norm of each token-head vector and the pairwise inner products of the difference vectors. After loading, norm errors stay within 1% and Gram-matrix errors within 10%.

The propagation analysis fixes tokens to separate hidden-state changes from changes in generated history. At downstream position i and layer l, define the relative key and value differences by

$$\varrho_{l,i}^K = \frac{\|k_{l,i}^S - k_{l,i}^I\|_2}{\|k_{l,i}^I\|_2}, \qquad \varrho_{l,i}^V = \frac{\|v_{l,i}^S - v_{l,i}^I\|_2}{\|v_{l,i}^I\|_2}, \tag{34}$$

for nonzero reference vectors. Layer summaries take medians over positions in the prompt after the reused span or in the response.

Both models retain a reduced key difference after the intermediate-layer peak. For the prompt after the reused span, Qwen's key difference peaks at 0.218 in layer 24 and reaches 0.095 at layer 39. Ministral's corresponding values are 0.190 in layer 19 and 0.144 at layer 39. Matched-norm noise reaches 0.399 and 0.251 at layer 39, respectively. Therefore, the final-layer deviations depend on the direction of the original difference.

At matched initial norms, the structure-preserving control changes task outcomes in both models, and matched-norm noise changes them in Qwen. Table 7 reports task success and final-layer key and value differences on the 48 tool-schema pairs per model, and Figure 5 shows the full layer profiles.

Table 7: Direction controls and residual KV differences on the 48 tool-schema pairs per model.

	Qwen3-14B			Ministral-3-14B		
KV condition	Success	Key diff.	Value diff.	Success	Key diff.	Value diff.
Stale KV	48/48	0.095	0.236	48/48	0.144	0.304
Matched-norm noise	0/48	0.399	1.059	48/48	0.251	0.559
Structure-preserving control	0/48	0.544	1.325	0/48	0.583	1.111

Invalid endpoints count as failures. Key and value differences are median per-position relative norms at layer 39, over prompt positions after the reused span.
Matched-norm noise preserves the combined key and value norm of each layer. The structure-preserving control also preserves the norm of each token-head vector and the pairwise inner products of the difference vectors. Equation 34 defines the relative propagation differences.

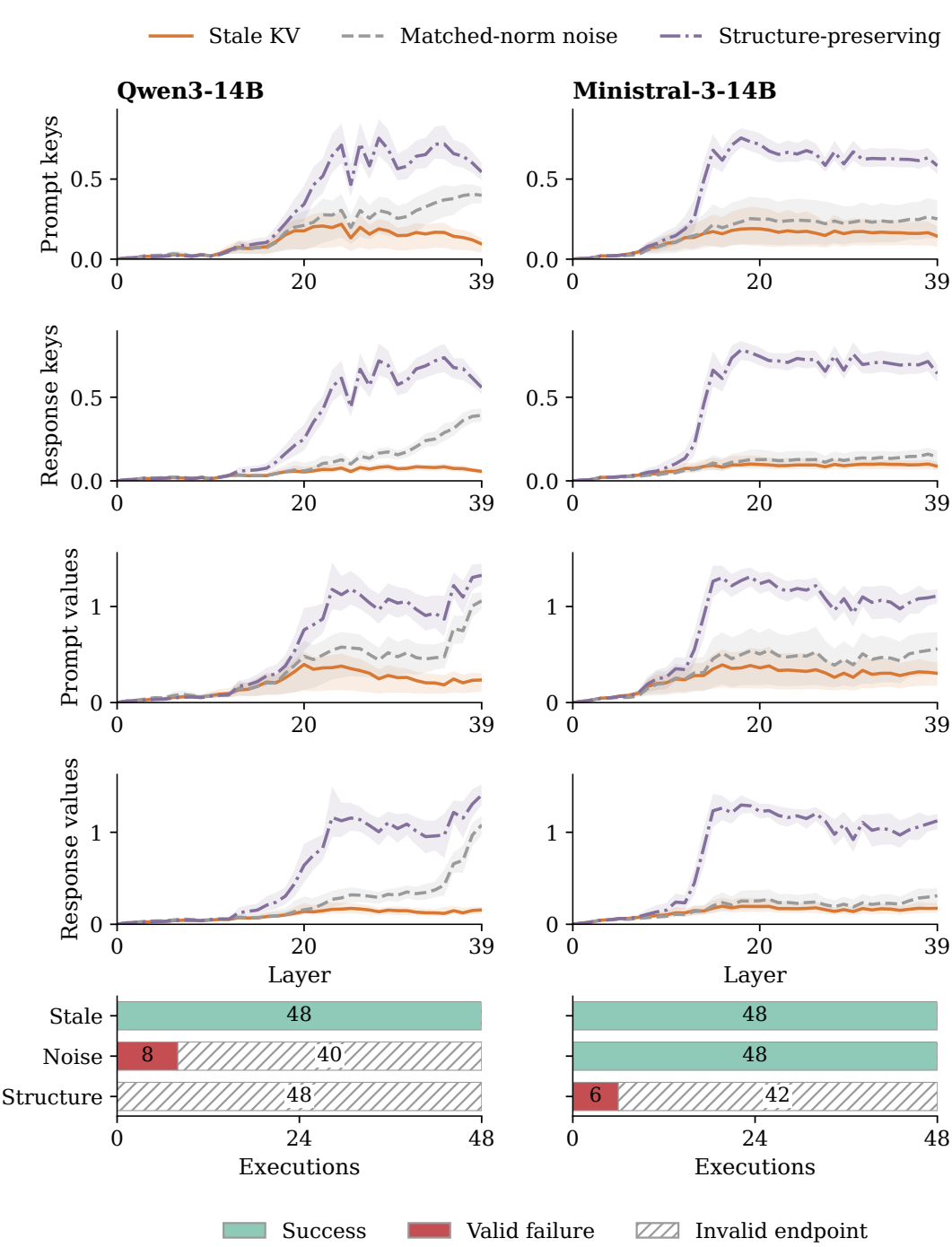

Figure 5: **Complete key and value propagation with direction controls.** Each model uses the 48 tool-schema pairs of Table 7 under fixed token sequences. Rows show relative deviations in prompt keys, response keys, prompt values, and response values across all 40 layers. Prompt positions follow the reused span. Lines are pooled-position medians, and bands span the 25th to 75th percentiles. Scales match across models within each row. Both direction controls match initial layerwise norms, and the structure-preserving control also keeps the norm of each token-head vector and the pairwise inner products of the difference vectors. Bottom bars split the corresponding 48 executions into successes, valid failures, and invalid endpoints. Hatched bars mark runs that exhaust their output budget without a valid endpoint, and these count as failures. This happens in 40 Qwen matched-norm noise runs, in all 48 Qwen structure-preserving runs, and in 42 Ministral structure-preserving runs.

Dose and localization. Eight tool-schema tasks per model receive the current KV plus a scaled stale difference or matched-norm noise at the same scaled norm. Other runs restrict the stale difference to keys, values, or one third of the layers. Table 8 reports task outcomes and the timing of the first token change. Every nonzero dose changes tokens in nearly every task. In Qwen, larger doses move the first change into the first request, and at twice the norm every first change in both models occurs there. Qwen tolerates the stale difference up to its full norm but noise only up to a quarter of it. Ministral tolerates both up to the full norm. Both directions fail at twice that norm. Keys alone, values alone, and each third of the layers change tokens in nearly every task, and every task succeeds except one Qwen keys-only execution that repeats an action. Divergence thus needs only part of the difference, and the tolerated magnitude depends on its direction.

Table 8: Scaled and localized KV differences on eight tool-schema tasks per model.

	Qwen3-14B				Ministral-3-14B			
Loaded difference	Success	Valid endpoint	No change	First request	Success	Valid endpoint	No change	First request
None (current KV)	8/8	8/8	8	0	8/8	8/8	8	0
Stale difference								
$\times 0.25$	8/8	8/8	1	1	8/8	8/8	0	7
$\times 0.5$	8/8	8/8	0	1	8/8	8/8	0	8
$\times 1$	8/8	8/8	0	6	8/8	8/8	0	7
$\times 2$	0/8	1/8	0	8	0/8	0/8	0	8
Matched-norm noise								
$\times 0.25$	8/8	8/8	2	1	8/8	8/8	0	4
$\times 0.5$	6/8	8/8	0	6	8/8	8/8	0	5
$\times 1$	0/8	1/8	0	8	8/8	8/8	0	5
$\times 2$	0/8	0/8	0	8	0/8	2/8	0	8
Stale keys only	7/8	7/8	0	3	8/8	8/8	0	7
Stale values only	8/8	8/8	0	8	8/8	8/8	0	8
Stale KV in one third of the layers								
First third	8/8	8/8	0	1	8/8	8/8	0	4
Middle third	8/8	8/8	0	5	8/8	8/8	0	8
Last third	8/8	8/8	1	0	8/8	8/8	0	5

Doses scale the stale-minus-current difference of every layer. Noise at each dose matches the realized key and value norm of the stale dose in each layer, with a fixed direction per task, layer, and component. No change counts executions without any token change, and first request counts executions whose first token change occurs in the first request. Invalid endpoints exhaust the output budget, except one Qwen keys-only execution stopped in a repeated-action loop, and count as failures.

C.2 Native Logits at Shared Positions

The native sampler records the full-vocabulary logits that select each generated token. We run every controlled task under recomputation, current KV, and stale KV with one seed, which gives 216 executions per model. Replays reproduce the original requests and outputs in all 216 Qwen executions and in 214 Ministral executions. The other two Ministral executions differ only in a directory timestamp of a tool observation, after their first token change. All 216 executions per model enter the comparisons, which use the logits recorded in these runs. Each comparison follows two executions of a task while their prompts and generated prefixes agree, up to their first token change or their end. Table 9 and Figure 6 report the four comparisons.

Every token change occurs at an at-risk position, and no change occurs where Equation 5 holds, as Proposition 1 requires. Every new token trails the current one by at most $2\|\delta_i\|_\infty$, as Corollary 1 requires. The bound certifies 99.6% and 92.2% of the unchanged Qwen positions and 83.0% and

65.6% of the unchanged Ministral positions. A token changes at 3.5% to 12.6% of at-risk positions across the four comparisons, while the at-risk share ranges from 0.4% to 37.5%. In the current-versus-stale comparison, margins below one occur at 1.0% of Qwen's and 11.0% of Ministral's shared positions. The newly selected token trailed by less than one in 40 of 70 and 58 of 72 first changes, respectively. The new token is the runner-up in 65 of 70 and 56 of 72 of them, and in 3 of 3 and 67 of 71 changes under current-KV loading.

Table 9: Native logits at shared generation positions of the 72 controlled tasks.

	Qwen3-14B		Ministral-3-14B	
Quantity	Recomputation vs. current	Current vs. stale	Recomputation vs. current	Current vs. stale
Executions with a token change	3/72	70/72	71/72	72/72
Executions without an at-risk position	28	0	0	0
Shared positions	23,226	8,303	5,338	1,530
At-risk positions (%)	86 (0.4)	710 (8.6)	965 (18.1)	573 (37.5)
Token changes at at-risk positions	3/3	70/70	71/71	72/72
Token changes per at-risk position (%)	3.5	9.9	7.4	12.6
Unchanged positions certified (%)	99.6	92.2	83.0	65.6
Smallest margin $m_i^\star$, median	20.38	20.75	9.00	8.53
$\|\delta_i\|_\infty$, median [90th percentile]	0.00 [0.00]	2.56 [5.59]	0.94 [1.81]	1.75 [3.62]
Shared tokens before the first change, median	28	74	18	7
Positions with $m_i^\star < 1$ (%)	345 (1.5)	87 (1.0)	455 (8.5)	169 (11.0)
First changes to the runner-up	3/3	65/70	67/71	56/72
First changes with a new-token margin below 1	3/3	40/70	69/71	58/72
New-token margin at the first change, median	0.25	0.75	0.12	0.38

Each column compares the two executions of every task up to their first token change or their end. A position is at risk when $2\|\delta_i\|_\infty \geq m_i^\star$, and it is certified when Equation 5 holds. The new token is the token the second condition selects at the first change, and its margin is how far it trailed the first condition's token. Margins and logit changes are in logit units of the native sampler, whose scale differs between the two models.

Margin and perturbation on shared inputs. We relate the observed token-change rate to $r_i = m_i^\star/(2\|\delta_i\|_\infty)$ for nonzero perturbations. For current versus stale KV, the four bins contain 60/172, 8/148, 2/390, and 0/7,593 changes in Qwen. Ministral gives 70/201, 2/126, 0/246, and 0/957. Figure 7 gives the bin boundaries and recomputation comparison. The observations stop at the first token change, inclusive, so the rates describe positions reached with a shared prefix.

The numerical baseline and stale-content comparison also use a common set of positions. Write R for recomputation, I for current KV, and S for stale KV. For each task, we compare R with I and I with S only while all three prompts and generated prefixes coincide. This yields 8,250 positions in Qwen and 1,168 in Ministral. We first average $\|\delta_i\|_\infty$ within each task, then give all 72 tasks equal weight. The stale-content increase over the numerical baseline is 3.05 [2.87, 3.23] in Qwen and 1.83 [1.42, 2.29] in Ministral. Brackets are 95% intervals from 2,000 paired bootstrap resamples of tasks. The increase is positive in 72 Qwen tasks and 70 Ministral tasks.

C.3 Persistence after Divergence

The continuation comparisons begin at the first different generated token. Let $\mathcal{K}_s^a$ be the retained span cache and w_b the first divergent token selected under condition b, for $a, b \in \{I, S\}$. Fixing the shared history and that token, then resuming free generation, defines

$$\tau^{a,b} = \tau(\mathcal{K}_s^a, w_b), \qquad (a, b) \in \{(I, S), (S, I)\}. \tag{35}$$

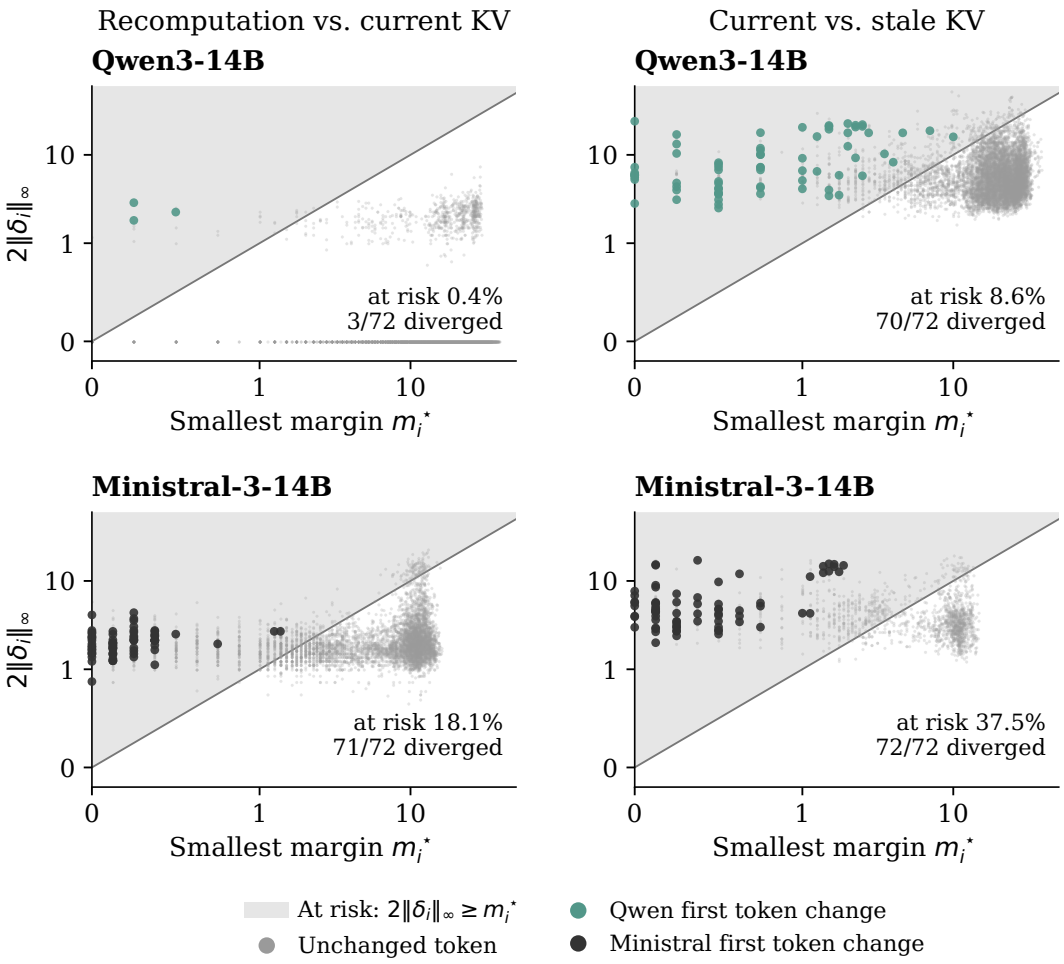

Figure 6: **Smallest margin against logit change at every shared position.** Each panel pools the shared positions of 72 tasks for one model and one comparison. Gray points are unchanged tokens, and green (Qwen) and black (Ministral) points are first token changes. The shaded region above the diagonal holds the at-risk positions, where $2\|\delta_i\|_\infty \geq m_i^\star$. Axes are logarithmic above 1 and use the logit units of each model. In the Qwen recomputation panel, 97% of positions have $\delta_i = 0$ and lie on the horizontal axis.

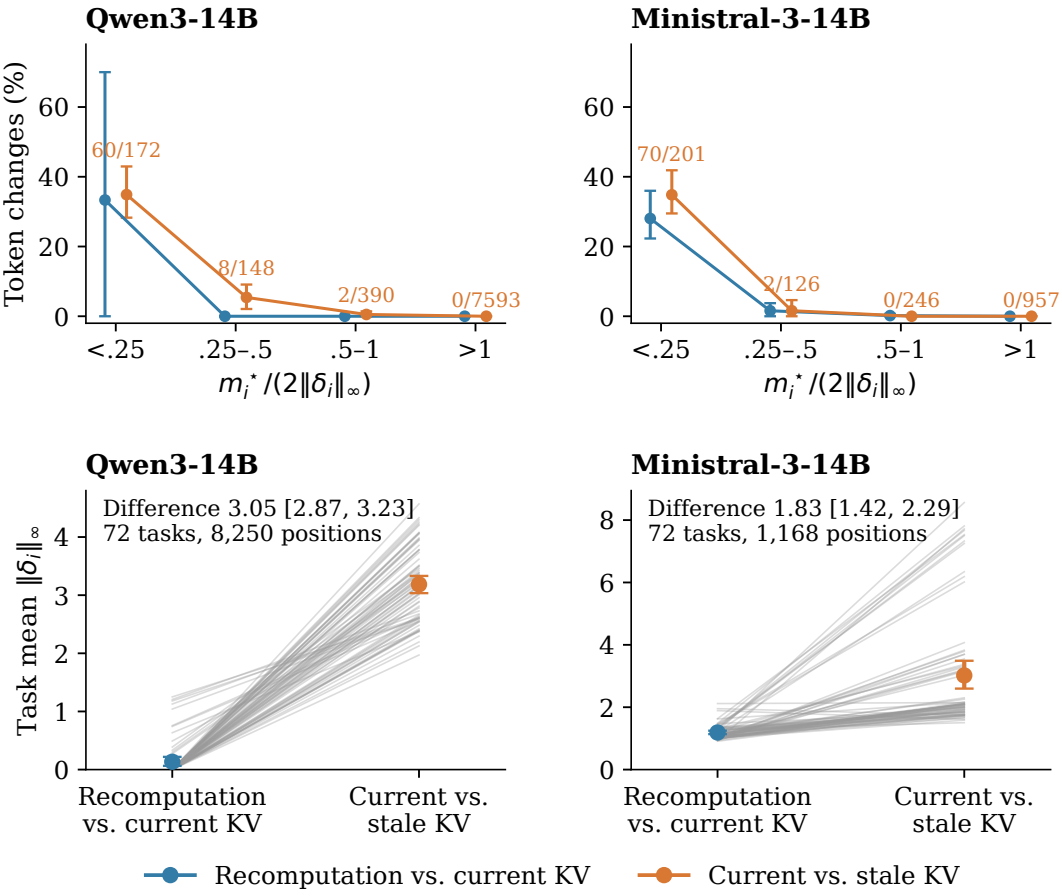

Figure 7: **Observed token sensitivity and a matched numerical baseline.** Top: token-change rates for nonzero score changes, with current-versus-stale counts above each point. The bins are $[0, 0.25)$, $[0.25, 0.5)$, $[0.5, 1]$, and $(1, \infty)$. Bottom: each gray line joins one task's mean perturbations on the same inputs. Colored points show means and 95% task-bootstrap intervals, using one seed and 72 tasks per model. Qwen's recomputation-versus-current comparison also contains 22,582 zero-perturbation positions, including 60 ties, with no token changes.

The crossed continuations prefill the shared response prefix and forced token, then resume greedy generation. The original executions τ^I and τ^S supply the reference action sequences. The crossed continuation $\tau^{I,S}$ uses current KV with the stale-selected token, and $\tau^{S,I}$ uses stale KV with the current-selected token. There are 280 crossed continuations for Qwen and 288 for Ministral. Two Qwen continuations have invalid behavior and are listed separately in Table 10. Each valid continuation, including unsuccessful ones, is classified by its action-sequence distances to the two original executions.

Table 10 separates the two exchanged-token conditions and their valid outcomes, and Figure 8 shows the action distances. Qwen has 176 valid continuations equally distant from the two original action sequences. Both models also contain continuations closer to the execution that supplied the token and others closer to the execution that supplied the KV. The crossed continuations can resemble either original action sequence.

Table 10: Task success and action distances after exchanging the first divergent token.

Model	KV	Forced token	Task success	Valid execution	Closer to token's path	Closer to KV's path	Equal distance
Qwen	Current	Stale	138/140	140/140	34	20	86
	Stale	Current	136/140	138/140	36	12	90
Ministral	Current	Stale	144/144	144/144	48	56	40
	Stale	Current	144/144	144/144	52	48	44

Every divergent pair produces two continuations with exchanged tokens (Equation 35). Success and validity use all collected continuations as their denominator. The final three columns partition valid continuations by action-sequence distance to the two original executions.

Replacing one token during native decoding. We select eight Qwen tasks by a fixed hash order among the 70 native-logit tasks with a current-versus-stale token change. Each task uses the registered seed and its original first divergence position. A greedy control leaves the logits unchanged at that position. The other conditions select the highest-scoring or second-highest-scoring non-greedy token from the current logits, breaking score ties by token ID. Each intervention changes only that token, then resumes ordinary greedy decoding and agent execution. The current span cache stays loaded under the original cache policy. The span and its preceding tokens match the original cache context in all 103 requests, while subsequent history evolves freely.

All eight greedy controls reproduce the original request inputs, output tokens, task outcome, stopping condition, and task files. Every alternative continuation also succeeds and has valid parsed responses (Table 11). Each alternative changes actions in six tasks and leaves different final task files in one task. Their median normalized action distances from greedy are 0.583 and 0.500, respectively. Successful alternatives therefore extend beyond the closest competing token in these tasks.

Table 11: A single token replacement during native decoding on eight Qwen tasks.

Token choice	Task success	Valid endpoint	Actions changed	Same task files	Current margin median [range]
Greedy	8/8	8/8	0/8	8/8	0
Runner-up	8/8	8/8	6/8	7/8	0.875 $[0.25, 2.5]$
Third-ranked	8/8	8/8	6/8	7/8	9.875 $[1.25, 13.25]$

Only the token at the original first divergence position is replaced. All eight greedy controls reproduce the original execution, and all responses parse in every condition. Action changes and task files are compared with the matched greedy control. File comparisons exclude runtime bytecode as in the original workspace metric. Margins use native logits.

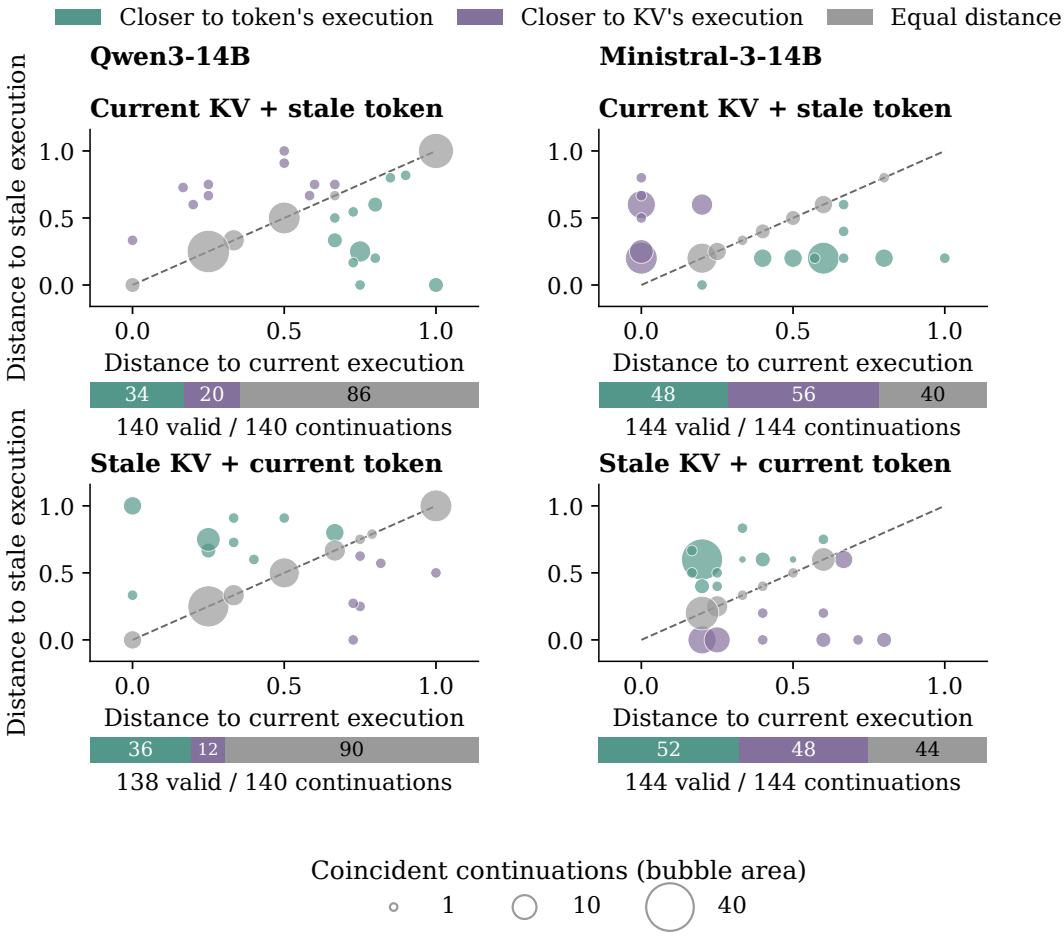

Figure 8: **Absolute action distances after exchanging the first divergent token.** Each continuation combines one original execution's KV with the other's first divergent token. Axes give action-sequence edit distances to the two originals, normalized by the longer sequence length. Bubble area is proportional to the number of coincident continuations. Colors and count bars show whether a continuation is closer to the execution that supplied the token, closer to the execution that supplied the KV, or equally distant. Qwen contributes 278 valid continuations, including four unsuccessful executions, with two invalid continuations excluded. All 288 Ministral continuations are valid. Rows show the two exchange directions separately.

Restoring one input after divergence. Eight tasks per model are continued from checkpoints one, two, and four turns after the first divergent turn. Each stale execution is continued in three ways: unchanged, with the current KV restored for the reused span in every later request, or with the current history restored under stale KV. All three keep the stale workspace. Table 12 compares each continuation with the current-KV continuation. No continuation returns to the current path, and all 72 per model succeed. In Qwen, restoring the KV shortens the median action distance from 10 to 6 after one turn, and restoring the history shortens it from 9 to 2.5 after two turns. In Ministral, restoring the history shortens it from 2 to 1, and restoring the KV does not shorten it.

Table 12: Restoring current KV or the current history after divergence.

		Qwen3-14B		Ministral-3-14B	
Turns	Continuation	Returns to current path	Action distance	Returns to current path	Action distance
1	Stale KV and stale history	0/8	10	0/8	2
	Current KV, stale history	0/8	6	0/8	2.5
	Stale KV, current history	0/8	9.5	0/8	1
2	Stale KV and stale history	0/8	9	0/8	2
	Current KV, stale history	0/8	7	0/8	2
	Stale KV, current history	0/8	2.5	0/8	1
4	Stale KV and stale history	0/8	7	0/8	1
	Current KV, stale history	0/8	5	0/8	1
	Stale KV, current history	0/8	6	0/8	1

Restoration applies after one, two, or four turns, counted from the first divergent turn. Action distance is the median tool-call edit distance, not normalized, to the current-KV continuation over eight tasks. All 72 continuations per model succeed, and every continuation keeps the workspace of the stale execution.

C.4 Absorption and State Rejoining

State rejoining compares observations and workspace snapshots at the same turn index. Observations are compared after removing runtime metadata such as process identifiers and working-directory paths. A rejoining turn requires equal observation and workspace-hash records at or after the divergence turn. Of Qwen's 140 divergent pairs, 36 rejoin in the divergence turn, 14 rejoin one turn later, and 90 do not rejoin before termination. Ministral has 144 divergent pairs, with 42 rejoining one turn later and 102 without rejoining. Of the pairs without rejoining, 76 of 90 in Qwen and all 102 in Ministral end in the same final workspace. Table 13 and Figure 9 give the distribution for each type of reusable text. All these pairs complete their tasks successfully. Most absorbed pairs in both models thus succeed without state rejoining.

C.5 Feedback and Task Success

Test-feedback interventions. We select eight tasks per model in a fixed hash order. Both executions must observe a failing test after their first token change, and each resumes from that test output under both caches. Full keeps the test output, Bit gives only success or failure, and Off removes it. Block also suppresses test reruns. The intervention leaves other tool calls and outputs unmodified, and both models use the same eight tasks. Every Full replay reproduces the original continuation exactly, and all 64 replays per model end validly (Table 14). Ministral succeeds in all 64 replays. Qwen succeeds in every Full replay. Without the test output, it fails three tasks with current KV and two with stale KV, and two tasks fail under both. With the bit alone, stale KV fails two tasks that current KV completes. The contrast $\Delta_{\mathrm{FB}}(\mathrm{Full}, \mathrm{Off})$ of Equation 24 is $-1/8$ for Qwen and zero for Ministral. Four Qwen replays under Bit leave changes outside the permitted file, and three still pass the checker (Figure 10). Therefore, removing test feedback changes current and stale success by similar amounts, and success without it can still violate an action requirement.

Table 13: Behavioral divergence and state rejoining by reusable text.

Model	Reusable text	n	Unchanged	Rejoins in divergence turn	Rejoins one turn later	No rejoining before termination
Qwen3-14B	Skill documents	48	0	20	4	24
	Tool schemas	48	0	8	4	36
	Policy instructions	48	4	8	6	30
Ministral-3-14B	Skill documents	48	0	0	26	22
	Tool schemas	48	0	0	4	44
	Policy instructions	48	0	0	12	36

Both cache conditions complete every listed pair successfully. The final three columns partition divergent pairs by observed state rejoining. Unchanged responses are reported separately from absorbed divergence.

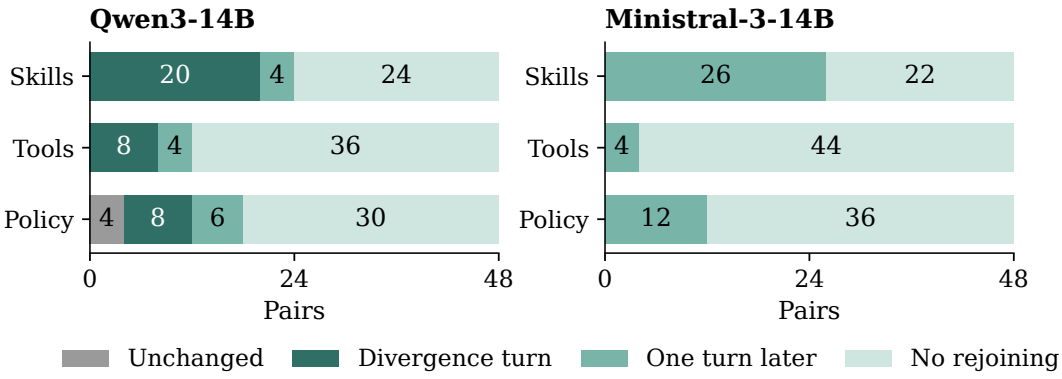

Figure 9: **Absorption and state rejoining across types of reusable text.** Each row contains 48 pairs that succeed under both caches. Skills, tools, and policy denote skill documents, tool schemas, and policy instructions. Bars separate unchanged responses from absorbed divergence that rejoins in the divergence turn, rejoins one turn later, or never rejoins before termination. Rejoining requires equal observations and workspace states. Counts include the four Qwen policy pairs without response divergence. Every type contains successful divergent executions without state rejoining.

Table 14: Task success after the first failing test output, by test-feedback condition.

	Qwen3-14B		Ministral-3-14B	
Test feedback	Current KV	Stale KV	Current KV	Stale KV
Full	8/8	8/8	8/8	8/8
Bit	8/8	6/8	8/8	8/8
Off	5/8	6/8	8/8	8/8
Block	5/8	6/8	8/8	8/8

Each cell counts successes over the same eight tasks, all with valid endpoints. Every Full replay reproduces the original continuation exactly.

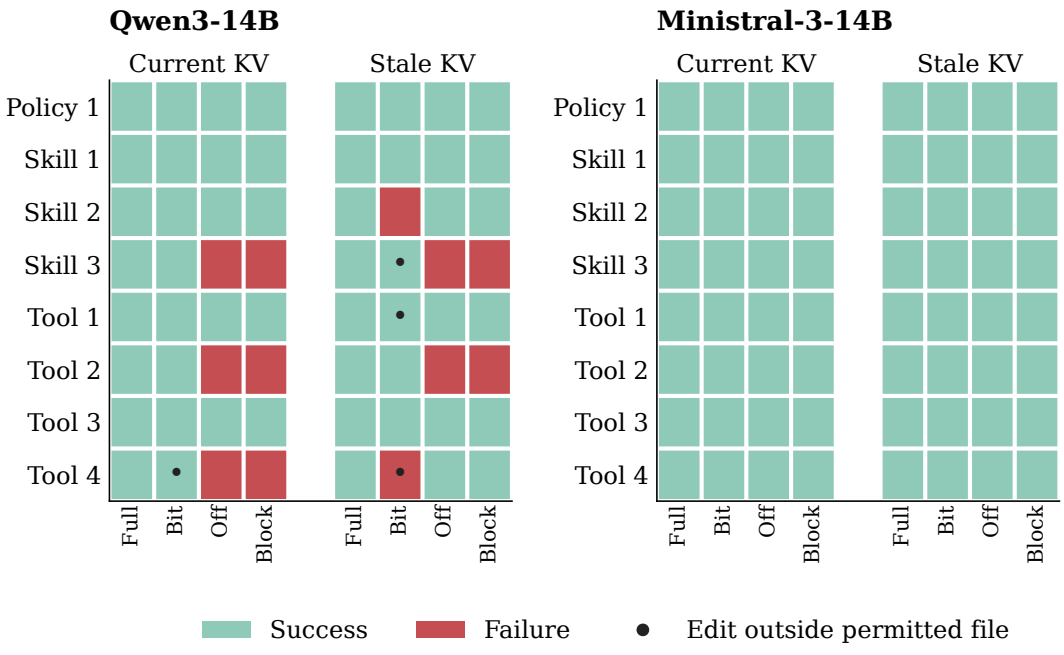

Figure 10: **Task-level outcomes under reduced test feedback.** Each row is one task, continued from its first failing test output after divergence. Both models use the same eight tasks. Columns show Full, Bit, Off, and Block under current and stale KV, and each cell represents one replay. Green cells succeed and red cells fail, and every replay ends validly. Dots mark replays that edit a file outside the permitted one.

Feedback observations. The Qwen feedback labels cover 140 divergent pairs. From the action divergence turn onward, 80 have no detected correction in either execution, 18 have equal positive counts, 22 have more detected correction with stale KV, and four have fewer. The remaining 16 have response divergence without action divergence, so they have no T_{act} from which to count feedback. In Equation 25, these 16 pairs have $U = 0$, and the 22 pairs with more detected correction with stale KV have $\nu = 1$. Among the 90 pairs without state rejoining, 80 have no detected correction and 10 have more detected correction with stale KV. Most absorbed Qwen divergences complete without detected correction.

C.6 Execution Costs and Reuse Assessments

Both models use the same pre-specified requirements. Qwen has 16 pairs with final workspace differences outside task checks and 12 other pairs exceeding process budgets. The remaining 116 pairs pass both checks and the action requirements. Ministral has matching final workspace states and passing process budgets in all 144 pairs. Table 15 and Figure 11 give the resulting assessments.

Table 15: Reuse assessments by our criterion for 144 valid pairs per model, all successful under both caches.

Model	Permissible reuse			Prohibited
	Transparent	Absorption-backed	Total	
Qwen3-14B	4 (2.8)	112 (77.8)	**116 (80.6)**	28 (19.4)
Ministral-3-14B	0 (0.0)	144 (100.0)	**144 (100.0)**	0 (0.0)

Counts (%). Execution budgets appear in Table 16.

Three kinds of checks implement B_z. Process caps bound the increases in output tokens, model turns, tool calls, failed actions, repeated actions, and parser corrections separately, and these six

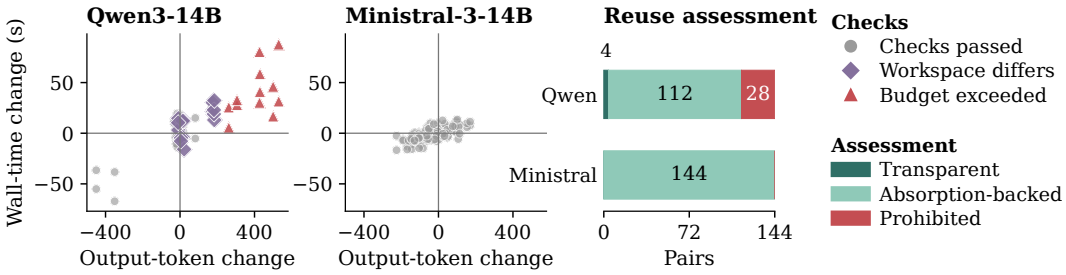

Figure 11: **Different execution costs despite task success.** Left and middle: stale-minus-current changes for 144 successful pairs per model, with markers for passed checks, workspace differences, and process-cap violations (Table 16). Wall time lies outside the process budgets. Output-token counts change in 134 Qwen pairs and all 144 Ministral pairs. Right: reuse assessments of the same pairs by our criterion (Table 15).

counts are the components of C_z (Table 16). Each count carries its own weight and budget, so $\lambda_C C_z$ and $\lambda_C \epsilon_C$ become weighted sums over the six counts, and Lemma 3 holds count by count. The workspace check requires final workspaces to match outside excluded runtime files, which gives $\Delta X_z = 0$. The measured action requirements check that responses parse and final file changes stay within the permitted set. Here X_z measures final workspace side effects and Q_z counts these recorded contract violations. All original executions have $Q_z = 0$.

Process distributions show the execution differences that medians hide (Figure 12). Table 16 reports means, medians, quartiles, and the shared budgets for paired execution changes.

Table 16: Paired execution changes and the shared process budgets.

		Qwen3-14B		Ministral-3-14B	
Quantity	Allowed increase	Mean	Median [quartiles]	Mean	Median [quartiles]
Output tokens	Pair-specific†	33.5	0.0 [−11.0, 8.5]	−31.7	−33.5 [−100.3, 29.3]
Model turns	2	0.36	0.0 [0.0, 0.0]	−0.07	0.0 [0.0, 0.0]
Tool calls	2	0.36	0.0 [0.0, 0.0]	−0.07	0.0 [0.0, 0.0]
Failed actions	1	0.00	0.0 [0.0, 0.0]	0.00	0.0 [0.0, 0.0]
Repeated actions	1	0.14	0.0 [0.0, 0.0]	0.00	0.0 [0.0, 0.0]
Parser corrections	0	0.00	0.0 [0.0, 0.0]	0.00	0.0 [0.0, 0.0]
Wall time (s)	Reported only	6.91	6.93 [−1.49, 12.50]	−1.28	−2.19 [−5.21, 1.89]

All entries summarize 144 stale-minus-current paired changes per model. Brackets give the 25th and 75th percentiles using linear interpolation. †The output-token allowance is the larger of 256 tokens and 25% of the current execution's output.
Workspace differences outside task checks are disallowed separately. The 16 workspace and 12 process-budget prohibitions in Table 15 do not overlap.

D Public-Benchmark Evaluation

SkillsBench tasks run in their released environments with their native verifiers. The 49 released SWE-Skills-Bench records carry no repository commits, so our harness rebuilds each repository at its last commit before the dataset's release and runs the released tests. Both models use 87 SkillsBench tasks (Li et al., 2026) and 49 SWE-Skills-Bench tasks (Han et al., 2026), with one shared seed per task across conditions and a 40,960-token context. Both models run with thinking disabled, and Qwen uses a 20-turn limit. Ministral uses a 500-turn limit, a time limit of twice each task's standard budget, and history compression above 30,000 tokens. Each task's skill text is the reused span. Its stale KV is computed for the same skill text placed after the instruction and environment of another task in the same benchmark. Divergence compares current and stale KV

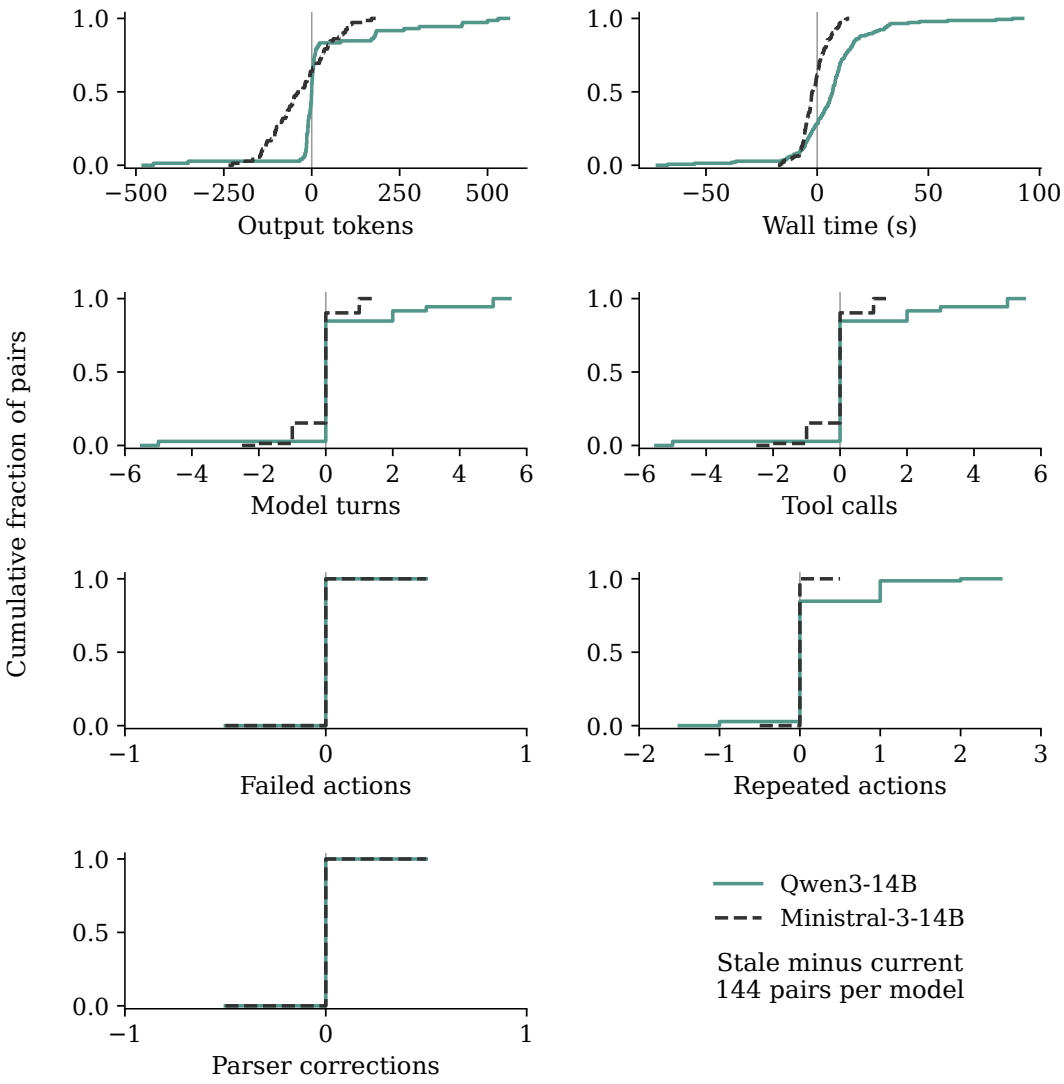

Figure 12: **Distributions of all recorded paired process changes.** Each curve gives the fraction of pairs whose change is at most the value on the horizontal axis. Each curve contains 144 pairs that succeed under both caches. Horizontal axes show stale-minus-current changes in the labeled metric. Wall time is measured in seconds and lies outside the process budget, and the remaining metrics are counts. Failed-action and parser-correction changes are zero in every pair.

cache reuse on the same task and initial visible input. Qwen's generated responses diverge on all 87 SkillsBench and all 49 SWE-Skills-Bench tasks, and its parsed actions on 85 and 47 of them. Ministral's responses and parsed actions diverge on every task in both benchmarks.

Table 17 reports tasks that pass under each condition and under pairs of conditions, and median stale-minus-recomputation changes in turns, tool calls, output tokens, and wall time over all tasks. Tasks without recorded test results count as not passing. Under recomputation, current KV, and stale KV, SkillsBench has 7, 7, and 7 such Qwen tasks and 9, 8, and 9 such Ministral tasks. SWE-Skills-Bench has 1, 1, and 2 such Ministral tasks.

Table 17: Tasks passing at least half of their tests, and execution costs, on the public benchmarks.

	SkillsBench		SWE-Skills-Bench	
Quantity	Qwen3-14B	Ministral-3-14B	Qwen3-14B	Ministral-3-14B
Tasks	87	87	49	49
Passing tasks, recomputation	7	46	24	41
Passing tasks, current KV	8	41	23	44
Passing tasks, stale KV	12	34	24	44
Passing tasks, current and stale KV	5	24	23	42
Passing tasks, recomputation and current KV	6	37	23	40
Passing tasks, recomputation and stale KV	6	29	23	39
Turn change	0	$+1$	0	0
Tool-call change	0	$+1$	0	0
Output-token change	-72	-365	$+72$	-522
Wall-time change (s)	$+22.65$	$+133.01$	$+35.73$	$+95.65$

A task passes when at least 50% of its recorded tests pass. Process changes are median stale-minus-recomputation differences over all tasks.