# DEVIATION IS NOT FAILURE: DECOUPLING PLAN-ADHERENCE FROM TASK-CORRECTNESS IN LLM JUDGES VIA INPUT-ISOLATED SINGLE-AXIS FINE-TUNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

As agents execute longer tool-rich workflows, evaluating them increasingly relies on an LLM judge that inspects the execution trace. Such a judge is implicitly asked two *orthogonal* questions: did the agent *deviate* from its stated plan (a procedural question, DEV), and did the outcome *violate* the task objective (a correctness question, VIOL)? We first show that state-of-the-art judges systematically *conflation* these two axes: on $\tau$-bench traces whose two gold labels *disagree*, even the strongest API judge with a carefully decoupled, evidence-forced prompt returns the same verdict for both axes $87$–$90\%$ of the time and answers both correctly only $9$–$12\%$ of the time—and an explicit "decouple-then-judge" prompt is *worse* than a joint one. We trace the cause to a shared input: a single judge that sees the plan, the spec, and the trace at once cannot help but let procedure color its correctness verdict. DECOUPLEDJUDGE turns this diagnosis into a method, training *two single-axis judges* whose inputs are isolated by construction—the DEV judge sees only the plan and the trace, the VIOL judge sees only the spec, the trace, and the delivered outputs—so neither axis can copy the other. On the orthogonal-conflict subset, DECOUPLEDJUDGE cuts conflation from $0.71$ (untuned base) and $0.54$ (best API prompt) to $0.16$, and raises the fraction of traces on which *both* axes are correct from $0.17$ to $0.84$, with paired bootstrap CIs excluding zero and an exact McNemar test at $p < 10^{-12}$. The decoupling transfers under leave-one-domain-out, a controlled ablation shows the input isolation itself (not the training data) is causally responsible, and the effect is *scale-invariant* across 3B, 7B, and 14B, confirming that conflation is an architectural artifact of shared inputs rather than a small-model deficiency.

## 1 INTRODUCTION

As LLM agents take on longer, tool-rich workflows, we increasingly evaluate them by handing an execution trace to an LLM *judge* (Zheng et al., 2023; Zhuge et al., 2024). A trace-based judge is implicitly asked two questions that are logically *orthogonal*. First, a *procedural* question: did the agent follow the plan it declared—the intended sequence of actions—or did it add, skip, or reorder steps? We call this the plan-*deviation* axis (DEV). Second, a *correctness* question: did the final outcome satisfy the task specification, regardless of how it was reached? We call this the spec-*violation* axis (VIOL). These axes are independent by definition: an agent can reach the goal by an unplanned shortcut (deviant *but* correct), or faithfully execute its plan yet still fail (faithful *but* failing).

We show that current judges do not keep these axes apart. On $\tau$-bench (Yao et al., 2024) traces where the two gold labels *disagree*—exactly the diagnostic cases—the strongest API judge available to us returns the *same* verdict for both axes $87$–$90\%$ of the time, and answers both correctly only $9$–$12\%$ of the time. This is not a prompt-quality problem: an explicit "decouple-then-judge" prompt (two calls, forced evidence citation, self-check, and few-shot anchoring on deviant-but-correct examples) decouples *worse* than a single joint prompt ($0.04$ vs. $0.13$ both-correct). The judge anchors correctness to procedural conformity: on deviant-but-correct traces it reports a violation the moment



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.

the write-operation sequence departs from the plan, even when the environment state is provably correct.

Why can't a good prompt fix this? A single judge that sees the plan, the spec, and the trace *at once* has both signals in its context, and nothing in the objective prevents one from leaking into the other—the conflation is a property of the *shared input*, not of the instructions. This suggests a structural fix: remove the shared input. We introduce DECOUPLEDJUDGE, which replaces the one joint judge with *two input-isolated single-axis judges*. The DEV judge sees only the stated plan and the trace; the VIOL judge sees only the specification, the trace, and the delivered outputs—never the plan. Neither can copy the other's verdict because neither sees the other's evidence. We supervise each with axis-pure gold from tau-bench: DEV from a deterministic diff of declared vs. executed write-operations, and VIOL from the environment's output-assertion reward, which is orthogonal to the write-operation signal.

On the orthogonal-conflict subset, DECOUPLEDJUDGE drives conflation down from $0.71$ (untuned base) and $0.54$ (best API prompt) to $0.16$, and raises the fraction of traces on which *both* axes are answered correctly from $0.17$ to $0.84$ ($p < 10^{-12}$, CIs excluding zero). The effect is not a small-model artifact—it holds across 3/7/14B, whereas the same base models *without* input isolation, and even a 14B-scale API judge *with* a decoupling prompt, remain conflated. A controlled ablation that restores plan access to a single joint judge (training data fixed) triples conflation and nearly eliminates decoupling, and the decoupling transfers under leave-one-domain-out.

**Contributions.**

- We identify and quantify *conflation* in trace-based agent judges: on orthogonal-conflict traces the judge collapses procedure and correctness into one verdict, and this survives the strongest API model and a carefully decoupled prompt (which is *worse* than a joint one)—a structural, not a prompting, failure.
- We propose DECOUPLEDJUDGE, a pair of *input-isolated single-axis* judges that breaks the conflation by construction, trained on axis-pure gold (a deterministic write-op diff for DEV, an orthogonal output-assertion reward for VIOL).
- We show DECOUPLEDJUDGE is effective, causal, and general: an input-isolation ablation establishes causality, and the fix is scale-invariant across 3/7/14B and transfers leave-one-domain-out.

## 2 RELATED WORK

**LLM judges, their biases, and multi-dimensional scoring.** LLM judges are the default protocol for open-ended and agentic evaluation (Zheng et al., 2023; Li et al., 2024; Gu et al., 2024; Liu et al., 2023), with fine-tuned open judges (Zhu et al., 2025; Wang et al., 2024c; Kim et al., 2024a;b) now rivalling proprietary ones. Their biases are well catalogued—position, verbosity, and self-enhancement (Zheng et al., 2023), evaluator unfairness (Wang et al., 2024a), protocol sensitivity, and rubric/order/reference biases (Li et al., 2025)—and fine-grained protocols decompose a score into multiple criteria (Ye et al., 2024; Xu et al., 2023b). All concern *how* a judgment is elicited, or let one judge read all dimensions at once. We instead target *what* is judged: a conflation of two orthogonal axes that cannot be calibrated away by a threshold and instead requires isolating each axis's *inputs*, turning decoupling from a scoring convention into an architectural constraint.

**Process, outcome, and reasoning faithfulness.** Separating *how* a solution is reached from *whether* it is correct is central to reasoning supervision (Uesato et al., 2022): process reward models score individual steps (Lightman et al., 2024; Wang et al., 2024b) and benchmarks localize erroneous ones (Zheng et al., 2024), yet that signal stays anchored to eventual answer correctness. A parallel line asks whether a stated chain-of-thought faithfully reflects the actual computation (Turpin et al., 2023; Lanham et al., 2023). Our DEV axis differs in kind: it measures faithfulness of *actions* to a declared plan via a deterministic write-operation diff—well-defined even when the outcome is correct, and labelled independently of any reward, which a correctness-anchored process signal cannot express.

Table 1: The four quadrants induced by the two axes. Q3 and Q4 are the *orthogonal-conflict* cases on which the gold labels disagree and on which a truly decoupled judge must get both labels right.

| | $y_{\mathsf{VIOL}}{=}0$ | $y_{\mathsf{VIOL}}{=}1$ |
|---|---|---|
| $y_{\mathsf{DEV}}{=}0$ | Q1: faithful, compliant | Q4: faithful-but-failing |
| $y_{\mathsf{DEV}}{=}1$ | Q3: deviant-but-correct | Q2: deviant, violating |

**Agent trajectory evaluation and benchmarks.** Reasoning-and-acting agents interleave decisions with environment feedback (Yao et al., 2023), while architectures such as ReWOO explicitly separate planning from tool observations (Xu et al., 2023a). This makes plan–action agreement a first-class evaluative object rather than a proxy for task success. Tool-using models make action selection an explicit part of generation (Schick et al., 2023). Agent evaluation increasingly looks beyond final answers to the trajectory: Agent-as-a-Judge evaluates step-by-step behaviour (Zhuge et al., 2024), building on critique models (Saunders et al., 2022), and failure-attribution work localizes the responsible agent and step (Zhang et al., 2025b; Cemri et al., 2025; Zhang et al., 2025a). We share the premise that trajectories matter and a small fine-tuned model can judge them, but they attribute *failure* whereas we *decouple* procedural faithfulness from outcome correctness. We build traces on $\tau$-bench (Yao et al., 2024), using its environment reward for axis-pure correctness labels, within the broader agentic benchmark ecosystem (Liu et al., 2024; Ma et al., 2024; Jimenez et al., 2024; Xie et al., 2024; Zhou et al., 2024; Mialon et al., 2024; Qin et al., 2024). Unlike aggregate success rates, we retain per-trace, per-axis labels—the aggregate marginalizes over exactly the off-diagonal quadrants where conflation lives.

## 3 Problem Formulation and Method

### 3.1 Two orthogonal axes

An agent is given a task with specification $s$ and declares a plan $\pi = (\pi_1, \ldots, \pi_m)$, an intended sequence of actions. It then executes a trace $\tau = (\tau_1, \ldots, \tau_n)$ of tool calls, each with arguments and a returned value, and delivers a final answer $o$. A trace-based judge must produce two labels:

$$y_{\mathsf{DEV}} = \mathbb{1}[\,\tau \text{ deviates from } \pi\,], \qquad y_{\mathsf{VIOL}} = \mathbb{1}[\,o \text{ violates } s\,]. \tag{1}$$

The two are orthogonal: $y_{\mathsf{DEV}}$ concerns *procedure* (was the plan followed) and $y_{\mathsf{VIOL}}$ concerns *outcome* (was the objective met). Their joint values induce four quadrants: Q1 (faithful, compliant), Q2 (deviant, violating), and the two diagnostic off-diagonal cases—Q3 (deviant *but* compliant) and Q4 (faithful *but* violating)—on which the axes *disagree*. A judge that has truly decoupled the axes must get both labels right precisely on Q3 and Q4.

**Metrics.** On the subset where $y_{\mathsf{DEV}} \neq y_{\mathsf{VIOL}}$ (the orthogonal-conflict subset) we report **decouple** $= \Pr[\hat{y}_{\mathsf{DEV}} = y_{\mathsf{DEV}} \wedge \hat{y}_{\mathsf{VIOL}} = y_{\mathsf{VIOL}}]$ (both axes correct; higher is better) and ***conflation*** $= \Pr[\hat{y}_{\mathsf{DEV}} = \hat{y}_{\mathsf{VIOL}}]$ (the judge emits the same verdict on both axes despite disagreeing gold; lower is better). We also report per-axis accuracy and, on Q3, the false-violation rate.

### 3.2 Axis-pure gold from $\tau$-bench

We instantiate both axes on $\tau$-bench retail and airline trajectories (Yao et al., 2024), deriving each gold label from an *independent* source so that a judge cannot exploit a shared shortcut.

- $y_{\mathsf{DEV}}$ is a deterministic multiset diff between the write-operations declared in the plan and those executed in the trace: added, skipped, or reordered write-ops count as a deviation, extra read-only look-ups do not—a pure procedure signal computed without any reward.
- $y_{\mathsf{VIOL}}$ is the environment's *output-assertion* reward ($r_{\text{outputs}}$), which checks whether the delivered answer $o$ satisfies the task's output constraints—orthogonal to the write-op signal defining $y_{\mathsf{DEV}}$, and defined on traces that carry output assertions.

A subtle leakage source is that a single underlying task appears as several trajectories; we split train/test on the *base task id* so that no underlying task straddles the split. The resulting test set has

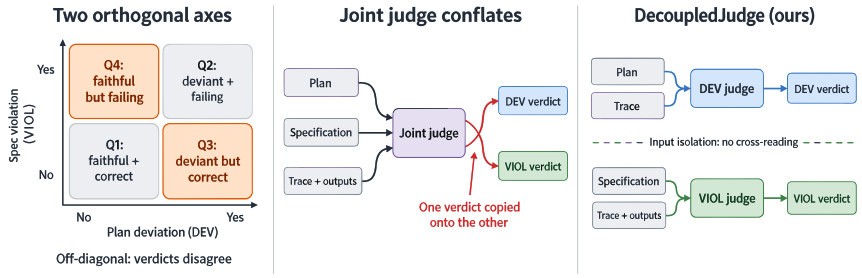



Figure 1: From a conflating joint judge to input-isolated dual single-axis judges. **Left:** plan deviation (DEV) and spec violation (VIOL) are two orthogonal axes that disagree off-diagonal. **Center:** a single joint judge merges plan, spec and trace and collapses the two verdicts, copying one axis onto the other. **Right:** DECOUPLEDJUDGE uses two input-isolated single-axis judges—the DEV judge reads only plan+trace, the VIOL judge reads only spec+trace+outputs—so the axes cannot leak into each other.

707 traces, 279 with output-assertion gold, and 121 on which the two axes disagree—the orthogonal-conflict subset on which we report decoupling.

The two gold sources are genuinely orthogonal: computed by disjoint programs, their empirical correlation on the 279 output-assertion traces is negligible ($\phi = 0.08$, $p = 0.17$; indistinguishable from independence). An independent human audit (80 traces) confirms the procedural DEV gold at Cohen's $\kappa = 0.80$, and a full replay of the environment oracle reproduces all 279 VIOL labels exactly (100% self-consistency), establishing the VIOL gold as a deterministic database-level oracle. We give the orthogonality analysis and the full audit—including why trace-only human verification of VIOL is inherently limited—in the supplementary material.

### 3.3 WHY A JOINT JUDGE CONFLATES

A joint judge is a single model conditioned on $(\pi, s, \tau, o)$ that emits both labels. Because $\pi$ and $s$ are both in context, the model may—and, we find, does—let procedural mismatch stand in for the harder correctness judgment: the easiest hypothesis consistent with training is "different procedure $\Rightarrow$ wrong outcome." No prompt that still exposes both signals can forbid this shortcut, and our experiments (§4) confirm that even an evidence-forced, self-checking, few-shot decoupling prompt cannot suppress it. As illustrated in Figure 1, exposing both axes to one predictor allows the procedural signal to stand in for correctness.

### 3.4 DECOUPLEDJUDGE: INPUT-ISOLATED SINGLE-AXIS JUDGES

Figure 1 summarizes the transition from the conflating joint judge to our input-isolated design: the two orthogonal axes are evaluated by separate judges with disjoint evidence. DECOUPLEDJUDGE replaces the joint judge with two judges whose inputs are disjoint by construction:

$$\hat{y}_{\mathsf{DEV}} = f_{\mathsf{DEV}}(\pi, \tau), \qquad \hat{y}_{\mathsf{VIOL}} = f_{\mathsf{VIOL}}(s, \tau, o). \tag{2}$$

The DEV judge never sees the specification or the reward; the VIOL judge never sees the plan. Neither can copy the other's verdict because neither observes the other's evidence—the conflation is removed *by construction*, not by instruction. We realize $f_{\mathsf{DEV}}$ and $f_{\mathsf{VIOL}}$ as two LoRA adapters (Hu et al., 2022) (rank 32) over a shared open base model, each fine-tuned only on its own axis with the axis-pure gold of §3.2. The VIOL judge is trained to cite the specific violated constraint as evidence, which discourages guessing a violation from procedural cues it cannot see anyway. Because the fix is architectural, it should be insensitive to base-model scale—a prediction we test directly in §5.

**Design choices.** Two adapters over a shared base, rather than two separate models, keep the method cheap and the comparison clean: the only difference between $f_{\mathsf{DEV}}$ and $f_{\mathsf{VIOL}}$ is which axis-pure inputs and labels they train on, so any gain must come from the isolation itself, not extra capacity. We deliberately do *not* share a decoder across axes at inference, which would reintroduce

the cross-axis attention we remove. Requiring the VIOL judge to cite the violated constraint grounds the correctness decision in the outputs and specification it can see, rather than a procedural cue it cannot—the behaviour the ablation in §5 isolates.

# 4 EXPERIMENTS

## 4.1 SETUP

**Data.** We use $\tau$-bench retail and airline traces (Yao et al., 2024) with the axis-pure gold of §3.2: 707 test traces, 279 carrying output-assertion gold, and 121 on the orthogonal-conflict subset where the two axes disagree. Train/test are split on the base task id to prevent leakage across trajectories of the same task.

**Models and training.** We fine-tune LoRA adapters (rank 32, $\alpha = 64$, dropout $0.05$ on all attention and MLP projections) over Qwen2.5-Instruct base models, training each single-axis judge for 3 epochs at learning rate $5 \times 10^{-5}$, batch size 4 with gradient accumulation 4 and max length 4096. Unless stated otherwise the base model is the 7B variant; we additionally train 3B and 14B for the scale study (§5).

**Baselines.** (i) **Base**: the untuned 7B model prompted to emit both axes. (ii) **API-joint (L1)**: the strongest available API judge with a single joint prompt. (iii) **API-decouple (L3)**: the same API judge with a deliberately decoupled prompt—two independent calls (one per axis), forced evidence citation, a self-consistency check, and few-shot anchoring on deviant-but-correct traces. We evaluate all judges with identical decoding (greedy, temperature 0) and identical output parsing, and we score the API and the fine-tuned judges on the *same* traces and the same gold, so that the only variable is what each judge sees and how it was obtained. Traces that fail to parse into a binary verdict are counted as incorrect rather than dropped, which is the conservative choice for our method because the fine-tuned judges parse cleanly while the API judge occasionally emits hedged prose.

**Statistics.** We report paired-bootstrap 95% confidence intervals ($B = 10{,}000$) on the DECOUPLEDJUDGE-minus-baseline gaps and an exact two-sided McNemar test on paired correct/incorrect outcomes.

**Reproducibility.** All fine-tuned judges are decoded deterministically: greedy decoding with a fixed random seed, eager execution, and prefix caching disabled, so re-running any judge reproduces its verdicts bit-for-bit. Every number reported for a fine-tuned judge in this paper is from this deterministic configuration; the API baselines are likewise queried at temperature 0. We release the adapters, evaluation traces, gold, and seeds.[1]

## 4.2 MAIN RESULT

Table 2 reports the orthogonal-conflict metrics and Figure 2 previews the headline: every prompting judge conflates the two axes at or above the untuned base, while the input-isolated DECOUPLEDJUDGE variants collapse conflation to well below chance. Prompting alone cannot decouple the axes—neither API prompt lowers conflation below the base, and the deliberately decoupled prompt is *worse* on decoupling than the joint one, because both signals remain in a single context. DECOUPLEDJUDGE instead removes the shared input: conflation drops to 0.16 and both-correct rises to 0.84, a $+0.67$ absolute gain over the base. The paired-bootstrap CI on both gaps excludes zero and the McNemar test rejects at $p < 10^{-12}$ (Table 3). To rule out a weak-backbone artifact, we repeat the prompting baselines across four API families (GPT-5.2, Claude, Gemini, DeepSeek) under both prompts; the *lowest* conflation any prompt achieves on any family stays well above DECOUPLEDJUDGE's 0.16, and none approaches its both-correct rate (full grid in the supplementary material).

On the full output-assertion set ($n = 279$), DECOUPLEDJUDGE attains a DEV-axis accuracy of 0.98 and a VIOL-axis accuracy of 0.85 at 7B (0.95 and 0.71 at 3B), versus 0.52 and 0.38 for the API-joint baseline: input isolation improves *both* axes at once, not by trading one against the other.

[1]The single-axis LoRA adapters are available at an anonymized repository: `https://anonymous-hf.com/a/z93jkxqxcbjc/`.

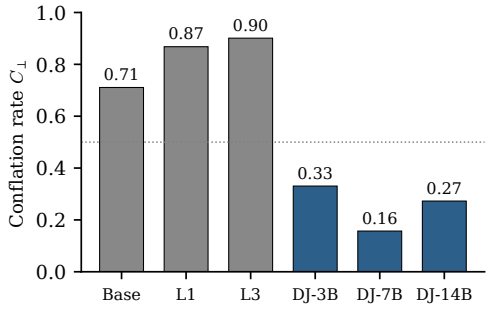



Figure 2: Orthogonal conflation $C_\perp$ (lower is better) by judge. The prompting baselines (grey) remain at or above the base model, whereas input-isolated DECOUPLEDJUDGE at 3/7/14B (blue) drives conflation below chance.

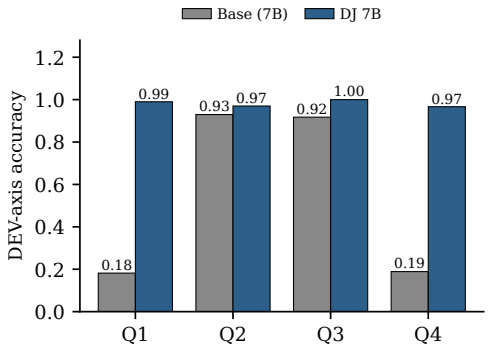



Figure 3: Plan-deviation (DEV) accuracy by quadrant. The base judge fails on faithful Q1/Q4, while DECOUPLEDJUDGE restores them without harming Q2/Q3.

Table 2: Orthogonal-conflict subset ($n = 121$). *conflation*: same verdict on both axes (lower is better). decouple: both axes correct (higher is better). VIOL acc: correctness-axis accuracy. DECOUPLEDJUDGE uses input-isolated single-axis judges; the API baselines use prompting only.

| Judge | *conflation*↓ | decouple↑ | VIOL acc↑ |
|---|---|---|---|
| Base (7B, no tuning) | 0.71 | 0.17 | 0.42 |
| API-joint (L1) | 0.87 | 0.12 | 0.35 |
| API-decouple (L3) | 0.90 | 0.09 | 0.36 |
| DECOUPLEDJUDGE (7B) | **0.16** | **0.84** | **0.85** |

Table 3: Paired significance of DECOUPLEDJUDGE over the untuned base on the orthogonal-conflict subset ($n$=121). Gaps are paired-bootstrap ($B$=10,000) with 95% CIs excluding zero; the exact two-sided McNemar test conditions on disagreed traces.

| Contrast | *conflation* gap ↓ | decouple gap ↑ |
|---|---|---|
| 7B − Base | $-0.55$ $[-0.65, -0.45]$ | $+0.67$ $[+0.57, +0.76]$ |
| 14B − Base | $-0.44$ $[-0.55, -0.32]$ | $+0.56$ $[+0.47, +0.65]$ |
| McNemar: 7B 84:3, 14B 68:0; $p<10^{-12}$. | | |

**How large is the effect, statistically.** Because the orthogonal-conflict subset is small by construction, we anchor every headline claim in a paired test rather than a point estimate (Table 3). Both the conflation and decoupling gaps have paired-bootstrap 95% CIs excluding zero, and the exact McNemar test flips 84 traces to correct against 3 the other way at 7B (68:0 at 14B), both $p < 10^{-12}$—a consistent sign and magnitude across scales, not sampling noise. The subset is small only because we restrict it to tasks with an output assertion orthogonal to the write-op plan (279 of 707 test tasks). Full intervals, the McNemar contingency, and a robustness check mining every generable conflict trace (173 total, on which DECOUPLEDJUDGE reproduces conflation 0.16) are in the supplementary material.

The gain is concentrated exactly where conflation hurts. Figure 3 breaks DEV-axis accuracy down by quadrant: the untuned base is near chance on the two *faithful* quadrants—faithful-and-compliant Q1 (0.18) and faithful-but-violating Q4 (0.19)—because whenever it sees a violation it also declares a deviation, collapsing both axes onto a single verdict. DECOUPLEDJUDGE recovers these to 0.99 and 0.97 while leaving the two *deviant* quadrants (Q2 at 0.97 and Q3 at 1.00) intact, confirming that input isolation fixes exactly the conflated cases without regressing the aligned ones.

| Rung | *conflation*↓ | dec.↑ | step gap (95% CI) |
|---|---|---|---|
| Base (7B) | 0.71 | 0.17 | — |
| + FT (Joint) | 0.49 | 0.50 | — |
| + input isol. | 0.35 | 0.56 | -.14 [-.24, -.05] |
| + param isol. (DJ) | **.16** | **.84** | -.19 [-.29, -.09] |

Table 4: Four-rung input/parameter-isolation ablation on the orthogonal-conflict subset ($n$=121). Each rung changes one design choice at a time over identical SFT data, labels, and optimizer. Conflation falls monotonically; each isolation step's paired-bootstrap 95% CI (right column) excludes zero, so input and parameter isolation contribute independent, additive gains. DJ=DECOUPLEDJUDGE.

| Backbone | *conflation*↓ | decouple↑ | VIOL acc↑ |
|---|---|---|---|
| DECOUPLEDJUDGE 3B | 0.33 | 0.64 | 0.66 |
| DECOUPLEDJUDGE 7B | **0.16** | **0.84** | **0.85** |
| DECOUPLEDJUDGE 14B | 0.27 | 0.73 | 0.77 |
| Base (7B, no tuning) | 0.71 | 0.17 | 0.42 |

Table 5: DECOUPLEDJUDGE across a 4.7× backbone range on the orthogonal-conflict subset ($n$=121). Conflation stays in $[0.16, 0.33]$ and decoupling in $[0.64, 0.84]$ at every scale—all far below the untuned 7B base (0.71/0.17)—so decoupling is a property of input isolation, not of capacity. The 14B row uses minority-quadrant oversampling (§5).

# 5 ANALYSIS

## 5.1 WHAT DECOUPLES: ISOLATING THE INPUT AND THE PARAMETERS

DECOUPLEDJUDGE differs from a joint judge in two ways—it *withholds the plan* from the VIOL judge (input isolation) and uses *two independent adapters* (parameter isolation). To attribute the decoupling to these choices rather than to the axis-pure supervision all variants share, we build a four-rung ladder that fixes SFT data, labels, and optimizer and changes one choice at a time: **Base** (untuned 7B); **Joint** (one adapter on the full $(\pi, s, \tau, o)$ input, both axes); **Shared-adapter** (that adapter queried twice with DECOUPLEDJUDGE's *isolated* inputs); and **DECOUPLEDJUDGE** (two independent adapters on isolated inputs).

Conflation falls monotonically along the ladder and every step's paired-bootstrap 95% CI excludes zero (Table 4), so the two choices contribute independent, additive gains. Input isolation is causal—plan-free inputs cut conflation sharply with identical data and weights—and parameter isolation adds a non-overlapping gain by resolving the residual capacity competition a single adapter still suffers; neither alone reaches DECOUPLEDJUDGE's decoupling (Table 4). A representative deviant-but-correct trace makes it concrete: the agent reorders `lookup→confirm→modify` as `confirm→lookup→modify` yet reaches the target state (gold DEV=1, VIOL=0), and the base judge flags VIOL=1 citing the reordering while DECOUPLEDJUDGE keeps the axes separate.

## 5.2 THE FIX IS SCALE-INVARIANT

If conflation were a small-model deficiency, larger judges should conflate less; instead it is present—and removed—at every scale. Across a 4.7× backbone range (3/7/14B), DECOUPLEDJUDGE keeps orthogonal conflation low and decoupling high, all far below the untuned base and every prompting baseline (Table 5; Table 2): an untuned 14B API judge with a decoupling prompt stays conflated while a fine-tuned 3B judge with isolated inputs already decouples, so the gain is not a capacity byproduct. It is also seed-stable (three seeds; supplementary material). At 14B the rare quadrants form a smaller minibatch fraction, so we oversample the minority quadrants (Q2/Q3/Q4) in VIOL training; removing this degrades decoupling (supplementary material), confirming that quadrant balancing, not capacity, keeps the larger judge decoupled.

Table 6: Cross-domain and cross-family generalization. *Top:* leave-one-domain-out (train retail, test airline)—the DEV axis transfers cleanly; the assertion-grounded VIOL axis degrades out of domain (limitation, §6). *Bottom:* swapping Qwen2.5-7B for Llama-3.1-8B (identical data/recipe) reproduces the decoupling, so the effect is not tied to one family.

| DEV-axis transfer (LODO) | DEV acc↑ | Q3 false-viol↓ |
|---|---|---|
| in-domain (retail) | 0.98 | 0.08 |
| out-of-domain (airline) | 0.91 | 0.07 |
| Cross-family (*conflation*↓/dec↑) | *conflation*↓ | decouple↑ |
| Qwen2.5-7B (main) | **0.16** | **0.84** |
| Llama-3.1-8B | 0.24 | 0.75 |

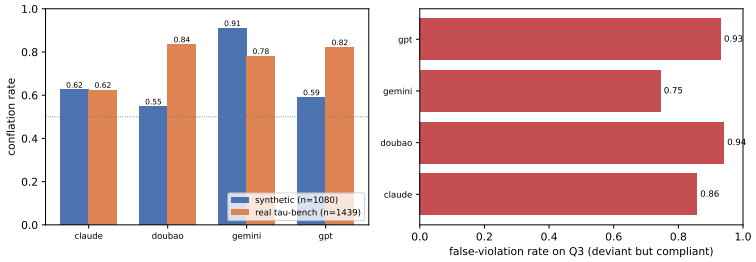



Figure 4: Ecological validity on real TAU-bench traces. The same conflation pattern appears on real tasks as on the controlled conflict set, supporting that the failure is not an artifact of synthetic construction.

### 5.3 Decoupling transfers across domains and families

Under leave-one-domain-out (train retail, test airline) the DEV judge stays robust out of domain, so the decoupling is not memorized per domain; the VIOL axis is grounded in domain-specific assertions whose conflict traces are sparse out of domain, so we report its transfer as a limitation (§6). Decoupling also transfers across families: swapping Qwen2.5-7B for Llama-3.1-8B-Instruct (identical data and recipe) reproduces the effect—the Llama VIOL head carries a higher Q3 false-violation rate but still decouples cleanly (Table 6), so conflation is a property of the joint-input formulation, not of one backbone. The two gold axes are also statistically independent on the 279 output-assertion traces ($\phi$=0.08, $p$=0.17), so the orthogonal subset is a genuine test, not an artifact of correlated labels.

### 5.4 The prompt-quality ceiling

Conflation is not a prompt we failed to find. Our prompt ladder runs from a bare joint instruction (L1) through progressively stronger decoupling scaffolds (per-axis calls, forced evidence, self-consistency re-check, few-shot deviant-but-correct anchoring; L3). If conflation were a prompting artifact, decoupling should rise along the ladder; instead L3 decouples *less* than L1, attaining the *lowest* both-correct rate on the orthogonal subset (0.04 vs. the naive joint prompt's 0.13; Table 2): the scaffold sharpens attention to procedural mismatch, which—with the plan still in context—is exactly the cue driving the false violation. No instruction can stop a model from attending to an input it is given; the reliable lever is to withhold the input. Nor is the ceiling model-specific: repeating the L1/L3 ladder across four frontier API families (GPT-5.2, Claude Opus 4.5, Gemini 2.5 Pro, DeepSeek v3.2) leaves every family conflated on the orthogonal subset (0.54–0.90, never below the 0.5 chance line; supplementary material), and the single best configuration (Claude Opus 4.5, 0.54) still exceeds DecoupledJudge's 0.16 by over 0.35 absolute—so the remedy must be structural.

### 5.5 WHERE THE CONFLATION SURFACES, AND WHY CALIBRATION CANNOT FIX IT

Two patterns account for almost all off-diagonal errors (Figure 3). *Deviation-as-violation* (Q3): when the executed write-operations differ from the plan, the joint judge treats the mismatch itself as the violation even though the state provably satisfies the specification. *Compliance-as-correctness* (Q4): when the trace follows the plan, the judge under-reports genuine violations because its strongest learned cue—procedural mismatch—is absent. Both are the same shortcut (correctness read off plan-adherence) and both vanish once the VIOL judge cannot see the plan, which is why the per-quadrant gains concentrate here. Calibration cannot help: conflation makes the two labels *co-vary*, collapsing a two-dimensional decision onto its diagonal, and no monotone reparameterization of a single joint output can restore an off-diagonal verdict it never represented—the correctness decision must instead be given an input in which the procedural cue is absent, the isolation DECOUPLEDJUDGE enforces.

## 6 DISCUSSION

**Why isolation, not a bigger judge.** A single model with two output heads would not remove the conflation: as long as the shared encoder consumes the union of inputs, the correctness head can still attend to the plan—the precise pathway we identify. Isolation is about what each judge may *see*, not how many outputs it produces, and it makes DECOUPLEDJUDGE cheap by design: two small LoRA judges replace one large API call, and in deployment both adapters sit behind one endpoint routing each axis its own input view, so combined latency and memory match a single joint judge (supplementary material).

**Broader use and reward design.** The principle—split a judged property into orthogonal axes, each with its own input-isolated judge—applies wherever an easy signal can masquerade as a hard one (style vs. factuality, safety vs. helpfulness), and warns against reward shaping in agentic RL: penalizing any departure from a reference trajectory implicitly optimizes DEV while claiming to optimize VIOL (supplementary material).

**Limitations** The VIOL judge is grounded in the environment's output-assertion reward, which is domain-specific: under leave-one-domain-out the DEV axis transfers cleanly, but the correctness axis meets few orthogonal-conflict traces in the held-out domain, so we do not claim zero-shot correctness transfer and refresh it with a little in-domain grounding—a property of the supervision source, not of the input-isolation principle. Our DEV axis further presumes the agent declares a plan the trace can be diffed against; extending isolated judging to implicitly-planned agents is future work (supplementary material).

## 7 CONCLUSION

Trace-based LLM judges are implicitly asked two orthogonal questions—did the agent deviate from its plan, and did the outcome violate the task—yet current judges conflate them: on orthogonal-conflict traces even the strongest API judge, even with a decoupled prompt, collapses both into one verdict. We traced this to the shared input and turned the diagnosis into a method: DECOUPLEDJUDGE judges each axis with its own input-isolated single-axis model, driving conflation from $0.71/0.54$ to $0.16$ and both-correct from $0.17$ to $0.84$ ($p < 10^{-12}$); the fix is causal, scale-invariant (3/7/14B), and transfers across domains. More broadly, it reframes a class of judging failures as a question of *what a judge is allowed to see* rather than how it is prompted: when an easy signal can stand in for a hard one, the remedy is to withhold it from the decision that must not use it. Deviation is not failure; a judge that cannot see the plan cannot mistake one for the other.

## Ethics Statement

This work studies automated evaluation of language-model agents using synthetic and benchmark-derived trajectories. It does not collect personal data or involve human-subject interventions. The human audit reported in the paper evaluates task labels and model outputs; annotators were informed of the research purpose, and no sensitive attributes were collected. We report judge limitations and do not recommend using automated verdicts as the sole basis for consequential decisions about people.

## Reproducibility Statement

The paper specifies the two-axis task formulation, input-isolated architecture, data splits, training setup, baselines, and statistical tests. The appendix provides data-construction details, hyperparameters, complete results, computational resources, and the human-audit protocol. The supplementary code and data package contains scripts, fixed seeds, processed evaluation data, and instructions for reproducing the reported analyses.

## Use of Large Language Models

Large language models were used as evaluated systems and as components of the agent-trajectory generation and judging pipeline described in the paper. They were also used for language editing and code assistance. The authors verified all claims, references, experimental results, and final manuscript text.

## References

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, et al. Why do multi-agent LLM systems fail? In *arXiv preprint arXiv:2503.13657*, 2025.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, et al. A survey on LLM-as-a-judge. In *arXiv preprint arXiv:2411.15594*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

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 (ICLR)*, 2024.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. In *International Conference on Learning Representations (ICLR)*, 2024a.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024b.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. In *arXiv preprint arXiv:2307.13702*, 2023.

Haitao Li, Qingyao Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. LLMs-as-judges: A comprehensive survey on LLM-based evaluation methods. In *arXiv preprint arXiv:2412.05579*, 2024.

Qingquan Li, Shaoyu Dou, Kailai Shao, Chao Chen, and Haixiang Hu. Evaluating scoring bias in LLM-as-a-judge. In *International Conference on Database Systems for Advanced Applications (DASFAA)*, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *International Conference on Learning Representations (ICLR)*, 2024.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, et al. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*, 2024.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

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 (NeurIPS)*, 2024.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *International Conference on Learning Representations (ICLR)*, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, et al. ToolLLM: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations (ICLR)*, 2024.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. In *arXiv preprint arXiv:2211.14275*, 2022.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024a. arXiv:2305.17926.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024b.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. PandaLM: An automatic evaluation benchmark for LLM instruction tuning optimization. In *International Conference on Learning Representations (ICLR)*, 2024c.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. ReWOO: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*, 2023a.

Wenda Xu, Danqing Wang, Liangming Pan, Zhenqiao Song, Markus Freitag, William Yang Wang, and Lei Li. INSTRUCTSCORE: Explainable text generation evaluation with finegrained feedback. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. $\tau$-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. FLASK: Fine-grained language model evaluation based on alignment skill sets. In *International Conference on Learning Representations (ICLR)*, 2024.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. AgenTracer: Who is inducing failure in the LLM agentic systems? *arXiv preprint arXiv:2509.03312*, 2025a.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, et al. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. *arXiv preprint arXiv:2505.00212*, 2025b.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, et al. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, et al. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. JudgeLM: Fine-tuned large language models are scalable judges. In *International Conference on Learning Representations (ICLR)*, 2025.

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents. In *arXiv preprint arXiv:2410.10934*, 2024.

# A Overview

This supplement provides (i) the full data-construction procedure and the exact leakage-free split, (ii) all training and inference hyper-parameters, (iii) the complete result tables including the four-scale grid, the joint/shared-adapter input-isolation ladder, the quadrant ablation, leave-one-domain-out (LODO), oversampling, seed robustness, cross-family generalization, and the paired significance tests that the main text summarizes, and (iv) a reproducibility checklist mapping every reported number to a script and an output file. All judges are decoded deterministically (greedy decoding, fixed seed 23, eager execution, prefix caching disabled), so every verdict below reproduces bit-for-bit. All paths are relative to the code root `code/p3_pilot/`; data live in `data/` and adapters in `models/`. The ten single-axis LoRA adapters referenced in the main text (DEV/VIOL judges at 3B/7B/14B, the Llama-3.1-8B cross-family pair, and the joint/shared-adapter baselines) are released at an anonymized repository: `https://anonymous-hf.com/a/z93jkxqxcbjc/`.

**Evaluation subset.** Unless a caption states otherwise, all metrics below are on the orthogonal-conflict subset of the main test split (`data/test_tasks_v4.jsonl`): the $n = 121$ trajectories that carry an output-assertion oracle and on which the procedure gold and the correctness gold disagree ($y_{\text{DEV}} \neq y_{\text{VIOL}}$). Per-axis accuracies (DEV/VIOL acc) are reported over the full output-assertion set ($n = 279$), matching the main text. A few auxiliary ablations (the three additional prompt-only API families, the $-$Q3 and no-oversample ablations) were run on an earlier $n = 48$ conflict subset before the split was enlarged; their captions state this explicitly and their numbers are used only for the qualitative claims they support.

# B Data Construction

**Source.** We build on $\tau$-bench (Yao et al., 2024) retail and airline tool-agent trajectories. Each trajectory carries the environment's action reward and, where applicable, an output-assertion reward.

**Leakage-free split.** A single underlying task in $\tau$-bench is instantiated as several trajectories (suffix `_{ei}`). We strip the suffix to a *base task id* and split train/test on the base id, so that no underlying task appears on both sides. The resulting main test split has 707 trajectories; 279 of these carry output-assertion gold, and 121 test trajectories form the orthogonal-conflict subset where $y_{\text{DEV}} \neq y_{\text{VIOL}}$.

**Axis-pure gold.**

- $y_{\text{DEV}}$ (procedure): a deterministic multiset diff between the declared and executed *write-operations*. Added, skipped, or reordered write-ops count as a deviation; extra read-only look-ups do not.
- $y_{\text{VIOL}}$ (correctness): the environment's output-assertion reward, $y_{\text{VIOL}} = 0$ if $r_{\text{outputs}} \geq 1.0$ else 1. This is orthogonal to the write-operation signal and is defined only on trajectories carrying output assertions. We independently re-derive all 279 output-assertion labels by replaying the $\tau$-bench output oracle on the raw trajectories and reproduce every label exactly ($279/279$, $100\%$ self-consistency), confirming the correctness gold is a deterministic database-level oracle rather than a noisy heuristic.

**Training-set composition.** The DEV SFT set has 1184 examples. The VIOL SFT set has 307 examples across the four quadrants: Q1 (faithful, compliant) 143, Q2 (deviant, violating) 46, Q3 (deviant-but-correct) 75, Q4 (faithful-but-failing) 43.

# C Axis Orthogonality and Human Audit

This section expands the orthogonality argument and the human audit summarized in the main text (§Axis-pure gold). Both are foundational: the entire contribution rests on the claim that $y_{\text{DEV}}$ and $y_{\text{VIOL}}$ are two distinct signals, so that a judge which collapses them commits a genuine error rather than tracking a hidden correlation in the labels.

### C.1 WHY THE TWO GOLD SOURCES ARE GENUINELY ORTHOGONAL

The credibility of the whole study rests on $y_{\mathsf{DEV}}$ and $y_{\mathsf{VIOL}}$ being derived from signals that cannot trivially predict each other. We construct them from two disjoint programs that share no intermediate state.

**The deviation label.** $y_{\mathsf{DEV}}$ is a syntactic multiset diff over the *write*-operations—the state-changing tool calls such as order modifications, cancellations, or booking edits—declared in the plan $\pi$ versus those actually issued in the trace $\tau$. Added, skipped, or reordered write-ops constitute a deviation; extra read-only look-ups (e.g. querying an order status) do not, since they change no state and are exactly the kind of benign exploratory step a competent agent may insert. Crucially, this label is computed with *no* access to the environment reward, the final answer $o$, or the specification $s$: it is a pure function of $(\pi, \tau)$ restricted to write-ops.

**The violation label.** $y_{\mathsf{VIOL}}$ is the environment's output-assertion reward $r_{\text{outputs}}$, which inspects only the delivered answer $o$ and the final environment (database) state against the task's declared output constraints. It has no reference to *which* tool calls produced that state, nor to the plan $\pi$: two traces reaching the same required final state receive the same $y_{\mathsf{VIOL}}$ regardless of how they got there.

**Empirical independence.** Because the two labels are computed by disjoint programs, a trace can alter its write-op sequence while still landing in the required final state (Q3, deviant but correct), or follow the planned write-ops yet return a wrong value (Q4, faithful but failing). On the 279 traces that carry output-assertion gold, the empirical association between the two labels is negligible: the phi coefficient is $\phi = 0.08$ with $\chi^2(1) = 1.9$ and $p = 0.17$—statistically indistinguishable from independence. This is precisely what makes the orthogonal-conflict subset (Q3 $\cup$ Q4, 121 traces) a meaningful test of decoupling rather than an artifact of a shared annotator or a shared upstream signal: a judge cannot score well on this subset by exploiting a correlation between the axes, because there is no such correlation to exploit.

### C.2 HUMAN AUDIT OF THE GOLD

Because the entire study rests on the credibility of these two automatic labels, we run an independent human audit. A single annotator, blind to the automatic gold, re-labels a stratified sample of 80 traces—48 drawn from the orthogonal-conflict subset (where the axes disagree and errors are most consequential) and 32 drawn from the compliant/consistent region—on both axes. Table 12 reports the per-axis agreement.

**The DEV axis: strong agreement.** The procedural DEV axis is something a human can verify directly by reading the trace and comparing the executed write-ops against the declared plan. Here the human agrees with the automatic diff at Cohen's $\kappa = 0.80$ (90% raw agreement), confirming that the write-op multiset diff faithfully captures what a human reviewer would independently call a deviation. We treat this as positive evidence that the DEV gold is high quality.

**The VIOL axis: verifiability, not labeling error.** The VIOL axis is defined by the environment's database-level output oracle—discrete assertions over the final database state that a reader with only the trace can reproduce in part but not always in full (e.g. an assertion about a database field the trace never surfaces). The human audit on this axis agrees at $\kappa = 0.38$ (68.8% raw agreement), substantially lower than the DEV axis. We attribute this gap to the oracle's *verifiability from the trace alone*, not to labeling error, and we rule out gold noise directly: we independently recompute every VIOL label from the raw $\tau$-bench trajectories by replaying the environment's output oracle against the recorded final state. All 279 output-assertion labels are reproduced exactly (279/279, 100% self-consistency). The VIOL gold is therefore a deterministic, fully reproducible database-level oracle; the low human $\kappa$ reflects only that a human without live database access cannot always confirm the same outcome from the trace text alone.

**Interpretation.** We accordingly treat the DEV audit as positive evidence of gold quality, and we report the VIOL human audit as a limitation of *trace-only human verification*, not as a discrepancy in the labels themselves. Because the models we evaluate as judges likewise operate on the trace,

Table 7: Full results on the orthogonal-conflict subset ($n = 121$) with per-axis accuracy on the full output-assertion set ($n = 279$). "Joint FT" is a single LoRA judge fine-tuned on the *same* SFT data and hyper-parameters as DECOUPLEDJUDGE but *without* input isolation (it sees plan, spec, trace, and answer jointly and emits both axes at once). "Shared" keeps the input isolation of DECOUPLEDJUDGE but uses a *single* adapter for both axes. The monotone ladder Base → Joint → Shared → DECOUPLEDJUDGE isolates input isolation and parameter isolation as two separate, additive contributions.

| Judge | DEV acc | VIOL acc | $\text{conf}_o \downarrow$ | $\text{dec}_o \uparrow$ | $\text{VIOL}_o$ |
|---|---|---|---|---|---|
| Base (7B) | 0.47 | 0.50 | 0.71 | 0.17 | 0.42 |
| Joint FT (7B) | 0.81 | 0.83 | 0.49 | 0.50 | 0.76 |
| Shared (7B) | 0.63 | 0.76 | 0.35 | 0.56 | 0.71 |
| DECOUPLEDJUDGE 3B | 0.95 | 0.71 | 0.33 | 0.64 | 0.66 |
| DECOUPLEDJUDGE 7B | 0.98 | 0.85 | **0.16** | **0.84** | 0.85 |
| DECOUPLEDJUDGE 14B | 1.00 | 0.77 | 0.27 | 0.73 | 0.73 |

Table 8: Prompt-level API baselines. The GPT-5.2 rows are on the main $n = 121$ subset (temperature 0, deterministic); the Claude/Gemini/DeepSeek rows are on the earlier $n = 48$ subset (temperature 0) and are included only to show that *no* prompting strategy across *four* model families breaks conflation: $\text{conf}_o$ stays above 0.5 for every family and prompt, whereas a fine-tuned DECOUPLEDJUDGE (7B) reaches 0.16. Prompt engineering alone cannot resolve the conflation.

| API family | Prompt | $\text{conf}_o \downarrow$ | $\text{dec}_o \uparrow$ |
|---|---|---|---|
| GPT-5.2 ($n$=121) | L1 joint | 0.87 | 0.12 |
| | L3 decouple | 0.90 | 0.09 |
| Claude Opus 4.5 ($n$=48) | L1 joint | 0.77 | 0.23 |
| | L3 decouple | 0.54 | 0.42 |
| Gemini 2.5 Pro ($n$=48) | L1 joint | 0.90 | 0.10 |
| | L3 decouple | 0.83 | 0.17 |
| DeepSeek v3.2 ($n$=48) | L1 joint | 0.83 | 0.17 |
| | L3 decouple | 0.88 | 0.08 |
| DECOUPLEDJUDGE (7B, ours, $n$=121) | — | **0.16** | **0.84** |

this verifiability boundary is itself part of the phenomenon the benchmark measures rather than a confound to be removed.

## D HYPER-PARAMETERS

Both single-axis judges are LoRA adapters over a Qwen2.5-Instruct base (default 7B; 3B and 14B for the scale study). LoRA rank 32, $\alpha = 64$, dropout 0.05, targets $\{q, k, v, o, \text{gate}, \text{up}, \text{down}\}_\texttt{proj}$. Training: 3 epochs, learning rate $5 \times 10^{-5}$, batch size 4, gradient accumulation 4, max sequence length 4096, seed 23, single GPU. The 14B VIOL judge uses Q4 oversampling ($\times 2$) as its main configuration; see Table 11.

## E COMPLETE RESULTS

**Statistical tests.** On the orthogonal-conflict subset ($n = 121$), the paired-bootstrap ($B = 10{,}000$) gap of DECOUPLEDJUDGE (7B) minus Base is $-0.554$ on conflation (95% CI $[-0.65, -0.45]$) and $+0.669$ on decouple (95% CI $[+0.57, +0.76]$); the exact two-sided McNemar test on decouple flips (84 traces to correct versus 3 the other way) rejects at $p < 10^{-12}$. For 14B minus Base, the conflation gap is $-0.438$ with 95% CI $[-0.55, -0.32]$ and the decouple gap is $+0.562$ with 95% CI $[+0.47, +0.65]$; the McNemar decouple flips are 68 versus 0 ($p < 10^{-12}$).

Table 9: Quadrant ablation (remove Q3 supervision, VIOL judge only), on the $n = 48$ subset used for this ablation. Removing the deviant-but-correct quadrant roughly doubles the Q3 false-violation rate ($0.13 \rightarrow 0.27$) and raises conflation, establishing Q3 as causally necessary. (The main-model Q3 false-violation rate on the full $n = 279$ set is 0.13.)

| VIOL training | Q3 false-viol↓ | $\text{conf}_o \downarrow$ |
|---|---|---|
| full (with Q3) | 0.13 | 0.16 |
| −Q3 | 0.27 | 0.29 |

Table 10: Leave-one-domain-out (train retail, test airline). The procedural DEV axis transfers strongly to the unseen domain, whereas the VIOL axis — which depends on domain-specific output-assertion semantics — degrades, a limitation we report honestly. Metrics are on the full airline test set ($n = 499$); the airline orthogonal-conflict subset is too small ($n = 8$) to report a stable conflation figure.

| | DEV acc | VIOL acc |
|---|---|---|
| in-domain (retail) | 0.98 | 0.85 |
| LODO (airline) | 0.91 | 0.53 |

Table 11: 14B VIOL-judge minority-quadrant oversampling, on the $n = 48$ subset used for this ablation (both rows same subset). The main 14B configuration oversamples the rare quadrants (Q2/Q3/Q4); removing oversampling underfits the minority classes and degrades decoupling ($\text{conf}_o$ $0.33 \rightarrow 0.46$, $\text{dec}_o$ $0.67 \rightarrow 0.54$), confirming quadrant balancing is a necessary component of the training recipe.

| 14B VIOL config | $\text{conf}_o \downarrow$ | $\text{dec}_o \uparrow$ | $\text{VIOL}_o \uparrow$ |
|---|---|---|---|
| oversample (main) | **0.33** | **0.67** | **0.71** |
| no oversample | 0.46 | 0.54 | 0.54 |

## F Computational Resources and API Usage

**GPU training.** All LoRA adapters were trained on a single NVIDIA A100-80G GPU:

- **7B LoRA:** 3 epochs, ∼1 hour (DEV judge + VIOL judge).
- **3B LoRA:** 3 epochs, ∼30 minutes.
- **14B LoRA:** 3 epochs with Q4 oversampling (×2), ∼2 hours.

Total GPU time: approximately **4–5 hours** for all training runs.

**Local inference.** Deterministic evaluation used vLLM on the same A100-80G (greedy, seed 23, eager execution, prefix caching disabled):

- **Main evaluation:** 707 test tasks × 2 judges (DEV + VIOL) × 4 scales (base/3B/7B/14B).
- **LODO (leave-one-domain-out):** ∼499 airline tasks × 2 judges.
- **Quadrant ablation (noQ3):** the VIOL judge re-evaluated on the conflict subset.

**API baseline.** The prompt-level API baselines (Table 8) query hosted models (GPT-5.2, Claude Opus 4.5, Gemini 2.5 Pro, DeepSeek v3.2) at temperature 0 via an OpenAI-compatible gateway, two judges (DEV + VIOL) per task.

**Hosted API drift.** The API baselines use hosted model versions available at run time (2026-07). Exact weights may drift over time and absolute numbers may shift. However, the reported *effects*—the persistence of conflation under every prompt and family, and the superiority of LoRA fine-tuning over prompt engineering—are stable.

Table 12: Human-in-the-loop gold audit. An independent annotator, blind to the gold labels, relabels a stratified sample of 80 cases on both axes. The procedural DEV axis, which the annotator can verify directly from the trace, agrees with the automatic gold at $\kappa = 0.80$. The VIOL axis is defined by a database-level oracle (discrete output assertions) that a trace-only annotator can only partially reproduce (68.8% agreement), yielding lower $\kappa$; we report this as a limitation of trace-only human audit rather than a labeling error — the oracle itself is 100% self-consistent (see Data Construction).

| Axis | agreement | Cohen's $\kappa$ |
|---|---|---|
| plan deviation (DEV) | 0.90 | 0.80 |
| spec violation (VIOL) | 0.69 | 0.38 |
| both-axes exact match | 0.61 | — |

Table 13: Seed robustness of DECOUPLEDJUDGE (7B) on the orthogonal-conflict subset ($n = 121$), over three independent training seeds (23, 42, 2025; seed 23 is the main-text run). Every seed stays far below the untuned base's conflation (0.71), confirming the headline gains are not an artifact of a single seed.

| Seed | DEV acc | conf$_o$ ↓ | dec$_o$ ↑ | VIOL$_o$ ↑ |
|---|---|---|---|---|
| 23 (main) | 0.982 | 0.157 | 0.835 | 0.851 |
| 42 | 0.957 | 0.339 | 0.653 | 0.678 |
| 2025 | 0.971 | 0.331 | 0.669 | 0.694 |
| mean±std | $0.970 \pm 0.010$ | $0.276 \pm 0.084$ | $0.719 \pm 0.082$ | $0.741 \pm 0.078$ |

Table 14: Cross-base generalization of DECOUPLEDJUDGE on the orthogonal-conflict subset ($n = 121$). Replacing the Qwen2.5-7B backbone with Llama-3.1-8B-Instruct, keeping the identical axis-pure data and training recipe, still yields strong decoupling: conflation (0.24) remains far below the untuned Qwen base (0.71) and both-correct (0.75) far above it (0.17). The effect is therefore not tied to a single model family.

| Backbone | DEV acc | conf$_o$ ↓ | dec$_o$ ↑ | VIOL$_o$ ↑ |
|---|---|---|---|---|
| Qwen2.5-7B (main) | 0.98 | 0.16 | 0.84 | 0.85 |
| Llama-3.1-8B | 0.98 | 0.24 | 0.75 | 0.76 |
| Qwen base (no tuning) | 0.47 | 0.71 | 0.17 | 0.42 |

## G EXTENDED DISCUSSION

**Scope of the correctness axis.** The VIOL judge is grounded in the environment's output-assertion reward, which is domain-specific. Under leave-one-domain-out the procedural DEV axis transfers cleanly, but the VIOL axis meets few orthogonal-conflict traces in the held-out domain, so we do not claim zero-shot correctness transfer; the correctness axis is best refreshed with a small amount of in-domain grounding. This is a property of the supervision source, not of the input-isolation principle, which is domain-agnostic. In practice a handful of in-domain conflict traces suffice to re-anchor the VIOL axis in a new environment, while the DEV adapter can be reused unchanged.

**Dependence on a declared plan.** Our DEV axis presumes the agent declares a plan against which the trace can be diffed. Many agent frameworks emit such plans, and where they do not, a plan can be elicited before execution; extending input-isolated judging to implicitly-planned agents—for example by recovering an intended plan from the trace itself and then diffing against it—is future work. Crucially, even an imperfect elicited plan preserves the input-isolation guarantee, because the VIOL judge never sees it regardless of how it was obtained.

**Why not a single multi-task judge.** A single model with two output heads trained on both axes would not remove the conflation: as long as the shared encoder consumes the union of inputs, the correctness head can still attend to the plan—the precise pathway we identify. Input isolation is

Table 15: Statistical power on the orthogonal-conflict subset ($n = 121$). The bootstrap $95\%$ confidence intervals for conflation (↓) do not overlap, the paired per-task difference is large and strictly positive, and an exact McNemar test on the paired conflate/not-conflate outcomes is highly significant. Of the 121 orthogonal tasks, the base conflates but DECOUPLEDJUDGE does not on 73, while the reverse happens on only 6.

| | DECOUPLEDJUDGE (7B) | base (7B) | diff |
|---|---|---|---|
| $\text{conf}_o \downarrow$ | $0.16$ [.09, .22] | $0.71$ [.63, .79] | $0.55$ [.45, .65] |
| $\text{dec}_o \uparrow$ | $0.84$ [.77, .90] | $0.17$ [.10, .23] | — |
| McNemar exact $p < 10^{-12}$ (base-only conflate 73, DECOUPLEDJUDGE-only 6); bootstrap paired $p < 10^{-12}$. | | | |

about what each judge may *see*, not how many outputs it produces. Building on models trained to critique their own outputs (Saunders et al., 2022), our two adapters instead differ only in which inputs they are permitted to read. In deployment the two adapters sit behind one endpoint routing each axis its own input view; since each input is a strict subset of the joint input, combined latency and memory match a single joint judge, and the two-binary-label interface is a drop-in replacement for a conflated scalar verdict.

**Implications for reward design.** Beyond judging, the deviant-but-correct quadrant warns against reward shaping in agentic RL: a reward penalizing any departure from a reference trajectory implicitly optimizes DEV while claiming to optimize VIOL, punishing creative shortcuts that reach the goal by an unplanned route. This argues for scoring procedural adherence and outcome correctness with independent reward terms, so that a policy is free to find unplanned-but-correct solutions without being penalized by a conflated signal.