TOOLPLAN-BENCH: DIAGNOSING HIDDEN PLANNING FAILURES IN LONG-HORIZON TOOL-CALLING AGENTS BEYOND SUCCESS RATE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents are increasingly deployed on long-horizon, tool-calling tasks, yet they are almost universally evaluated by a single black-box outcome: task success rate. We argue that success rate systematically *hides* planning failures. An agent can reach the correct final state through a flawed plan—taking redundant actions, executing operations out of order, skipping required preconditions, or stopping prematurely on easy instances—and still be scored as a success. We introduce **ToolPlan-Bench**, a diagnostic benchmark that decomposes agent behaviour into a structured taxonomy of seven planning-failure modes and measures the **right-outcome-wrong-plan rate (ROWP)**: the fraction of *successful* trajectories that nonetheless carry a planning-failure label. Built on the real tasks, tools, and executable databases of τ-bench (retail and airline domains), ToolPlan-Bench evaluates six frontier models on 990 live agent trajectories. We find that ROWP is large and statistically robust—the bootstrap 95% lower bound exceeds 0.23 for every model, meaning at least one in four *successful* trajectories hides a planning flaw. ROWP rises monotonically with task complexity (from 0.43 to 0.57 as the number of required write operations grows), and 60% of trajectories diverge from the gold plan at the very first step. Five independent human annotators validate our rule-based labels (Cohen's $\kappa = 0.90$ between the human majority and the rule labels on the core label; $\kappa \geq 0.95$ on the structural dimensions underlying ROWP). Finally, we show that a natural intervention—prompting the agent to plan before acting—can significantly change success rate (in both directions across models) yet has *no significant effect on ROWP for any model*, demonstrating that the planning failures ToolPlan-Bench measures are robust to naive prompting and constitute an open challenge.

1 INTRODUCTION

Tool-calling LLM agents now underpin a growing range of applications, from customer-support automation to multi-step data workflows. To evaluate them, the community has converged on *task success rate*: whether the agent reaches a correct final state, typically verified by comparing the resulting environment or database against a gold reference (Yao et al., 2024; Qin et al., 2024; Liu et al., 2024; Guo et al., 2024). Success rate is attractive because it is objective, automatic, and end-to-end. But it is also a *black box*: it reveals only whether the outcome is right, not whether the agent got there by a sound plan.

We observe that this black-box view systematically hides an important class of failures. Consider an agent asked to modify a customer order. It may achieve the correct final database state—and thus be scored as a success—while along the way issuing redundant duplicate calls, performing a write before the read that should gate it, or skipping a required precondition that happened not to matter for this particular instance. The plan is wrong; the outcome is right. Success rate is blind to this. As tasks grow longer and involve more state-changing operations, such *right-outcome-wrong-plan* behaviour becomes both more likely and more consequential—a plan that succeeds by luck on an easy instance will fail on a harder variant.

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.

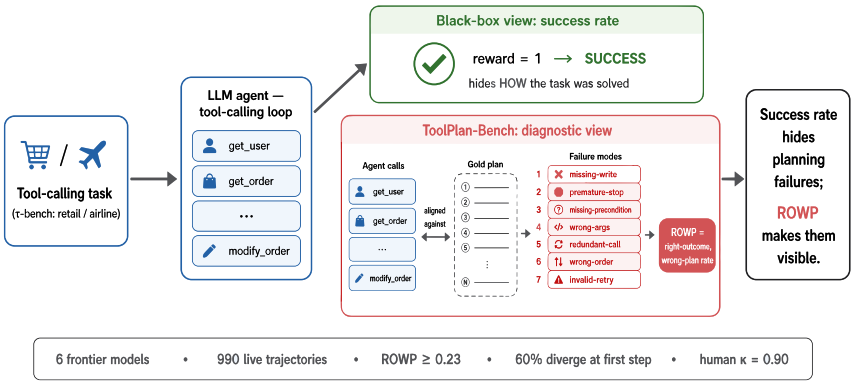

Figure 1: **Success rate hides planning failures; ToolPlan-Bench makes them visible.** An agent can reach the correct final state (*black-box view*: reward $= 1$, scored as success) while its trajectory contains planning flaws. ToolPlan-Bench aligns the agent's tool calls against a gold plan (*diagnostic view*), decomposes the trajectory into seven planning-failure modes, and measures the *right-outcome-wrong-plan rate (ROWP)*—the fraction of successful trajectories that still carry a planning flaw. Across six frontier models and 990 live trajectories, ROWP is large (≥ 0.23) and 60% of trajectories diverge from the gold plan at the very first step; a five-annotator human study yields $\kappa = 0.90$ against our rule labels.

ToolPlan-Bench. We introduce ToolPlan-Bench (Figure 1), a diagnostic benchmark designed to make these hidden planning failures visible and measurable. Rather than proposing yet another success leaderboard, ToolPlan-Bench decomposes each trajectory against its gold action sequence into a structured taxonomy of seven planning-failure modes (missing required write, premature stop, missing precondition, wrong arguments, redundant call, wrong order, invalid retry). Its headline metric is the **right-outcome-wrong-plan rate (ROWP)**: among trajectories that succeed (reward $= 1$), the fraction that still exhibit at least one planning-failure label. ROWP is precisely the quantity that success rate cannot see.

We instantiate ToolPlan-Bench on top of τ-bench (Yao et al., 2024), reusing its real customer-service tasks, executable tool APIs, and stateful databases in the retail and airline domains, and running six frontier models (gpt-5.2, claude-opus-4-8, gemini-3.5-flash, deepseek-v3, glm-5, qwen3.7-max) to produce 990 live agent trajectories. We deliberately use a single-turn formulation (the full instruction is given up front, without a user simulator) so that the entire burden of *planning* rests on the agent and can be diagnosed against the gold plan.

Findings. Our study yields four main results.

- **ROWP is large and robust.** Across models, the bootstrap 95% lower bound on ROWP exceeds 0.23: at least a quarter of *successful* trajectories hide a planning flaw, invisible to success rate.
- **ROWP grows with planning complexity.** Stratifying by the number of required write operations, ROWP rises monotonically from 0.43 (one write) to 0.57 (three or more), while 60% of trajectories diverge from the gold plan at the very first step—planning fails at the outset, not through gradual drift.
- **Models have distinct failure signatures.** claude is over-conservative (premature-stop, missing-write); gemini is redundancy-prone (redundant-call, wrong-args); deepseek skips steps (missing-write, missing-precondition); while qwen3.7-max, glm-5 and gpt-5.2 show flatter, more balanced profiles. A cross-domain strength flip (gemini excels on retail but collapses on airline, claude the reverse) further shows the benchmark's diagnostic power.
- **Planning failures resist naive fixes.** A plan-before-acting prompt significantly shifts success rate (up for claude/gpt-5.2, *down* for deepseek) yet leaves ROWP statistically unchanged for every model. The failures ToolPlan-Bench measures are not an artefact of prompting; they are a robust, open challenge.

We validate our rule-based labels with *five independent human annotators*, obtaining Cohen's $\kappa = 0.90$ between the human majority vote and the rule labels on the core planning-failure label and $\kappa \geq 0.95$ on all structural dimensions underlying ROWP. All tasks, tools, trajectories, diagnostic code, and seeds are released for reproducibility.

2 Related Work

Tool-calling and agent benchmarks. LLMs can learn to invoke APIs by self-supervision (Schick et al., 2023) or by connecting to massive real API pools (Patil et al., 2023; Song et al., 2023), and benchmarks probe whether/which tool to call (Huang et al., 2024; Li et al., 2023) and tool-use safety (Ruan et al., 2024). Interactive agent benchmarks span the web (Deng et al., 2023; Zhou et al., 2024), general assistants (Mialon et al., 2024), OS environments (Xie et al., 2024b), software engineering (Jimenez et al., 2024), app ecosystems (Trivedi et al., 2024), and long-horizon planning (Xie et al., 2024a). Among tool-calling evaluations, ToolLLM/ToolBench (Qin et al., 2024) scales tool learning to $16{,}000+$ real APIs and scores agents with ToolEval, which judges whether a solution path completes the task. StableToolBench (Guo et al., 2024) improves the stability of such large-scale tool benchmarks but likewise reports pass/win rates. AgentBench (Liu et al., 2024) evaluates LLMs as agents across eight environments and identifies long-horizon reasoning as a bottleneck, yet its reported metric remains task success. τ-bench (Yao et al., 2024) contributes realistic customer-service domains with executable tools, stateful databases, and a user simulator, scoring agents by comparing the final database state to a gold reference (with pass^k for reliability across trials). We build directly on τ-bench's tasks, tools, and databases, but depart from all of the above in *what* we measure: instead of whether the outcome is correct, we diagnose *how* the agent planned to get there, and expose failures that persist even among successful trajectories.

Process-level and analytical evaluation. Closest to our motivation is AgentBoard (Ma et al., 2024), which argues that binary success is too coarse and introduces a fine-grained *progress rate* to track partial advancement in multi-turn agent tasks. ToolPlan-Bench shares the premise that success rate is insufficient, but differs in kind. Progress rate is a scalar measure of *how far* an agent got; it does not decompose *why* a plan is flawed, and—critically—it says nothing about trajectories that fully succeed. ToolPlan-Bench instead provides a structured seven-mode taxonomy of planning failures and, through ROWP, targets exactly the trajectories that a scalar progress or success measure would mark as complete. The two views are complementary: progress rate diagnoses incomplete runs, ROWP diagnoses the hidden flaws inside completed ones. More broadly, the value of supervising the reasoning *process* rather than only the final outcome is well established in the math-reasoning setting: Uesato et al. (2022) contrast process- and outcome-based feedback and show that outcome supervision can leave reasoning errors uncorrected even when the final answer is right, and Lightman et al. (2023) demonstrate that step-level (process) supervision substantially outperforms outcome supervision, releasing dense human step-labels; subsequent work automates such step-level supervision to avoid costly human labels (Wang et al., 2024; Luo et al., 2024). ToolPlan-Bench carries this outcome-vs-process distinction into the *agentic tool-calling* setting: ROWP is precisely the rate at which an outcome-correct trajectory still contains a process (planning) flaw, a quantity outcome-only scoring cannot see.

LLM-as-a-judge and label validation. We validate our rule-based labels primarily against *five independent human annotators* (Section 4), reporting both inter-annotator Fleiss' κ and Cohen's κ between the human majority and the rule labels. Because scalable process-level annotation is expensive, we additionally run an independent LLM judge as a low-cost proxy, following a now-standard methodology: Zheng et al. (2023) show that strong LLM judges agree with human preferences at roughly the same rate humans agree with one another ($> 80\%$), while also cataloguing judge biases (position, verbosity, self-enhancement). LLM judges are now widely used both for open-ended generation (Liu et al., 2023; Kim et al., 2024) and for large-scale preference ranking (Chiang et al., 2024), though their reliability—especially sensitivity to answer position—remains under active scrutiny (Wang et al., 2023c). We treat neither source as ground truth but as *independent second annotators* that never see the rule labels and emit their own seven-dimension judgement. High agreement indicates that the rule labels capture what an independent annotator, given only the task, gold plan, and trajectory, would also call a planning failure.

Reasoning, planning, and acting agent methods. Prompting methods elicit better reasoning and planning from LLMs—chain-of-thought (Wei et al., 2022), self-consistency (Wang et al., 2023d), decomposition (Zhou et al., 2023; Wang et al., 2023b), thought search (Yao et al., 2023a), world-model planning (Hao et al., 2023), and as-needed planning (Prasad et al., 2024)—and agent frameworks build on them to compose tools and roles (Shen et al., 2023; Lu et al., 2023; Wu et al., 2024), act in open-ended worlds (Wang et al., 2023a; Park et al., 2023), and self-correct (Madaan et al., 2023; Gou et al., 2024). ReAct (Yao et al., 2023b) interleaves reasoning traces with actions and is the de facto backbone for tool-calling agents; our agent loop follows this paradigm. Reflexion (Shinn et al., 2023) adds verbal self-reflection to iteratively improve behaviour. These prompt-level improvements motivate our *partial-solution* study (Section 7), where we test whether an explicit plan-before-acting prompt—the most natural such intervention—reduces the planning failures ToolPlan-Bench measures. It does not, which positions these methods as insufficient for the challenge we surface rather than as baselines to beat.

3 The ToolPlan-Bench Benchmark

3.1 Task source and environment

ToolPlan-Bench reuses the tasks, tool APIs, and executable databases of τ-bench (Yao et al., 2024) in two domains: **retail** (115 test tasks, 16 tools) and **airline** (50 test tasks, 14 tools). Each task specifies a natural-language instruction, a gold action sequence (the reference plan), and a stateful database. Tools are genuinely executable against the database, and an outcome is scored by a deterministic reward $\in \{0, 1\}$ that compares the resulting database state (hash) against the gold final state. Tools divide into *read* operations (whose names start with `get_`, `list_`, `find_`, `calculate`, `think`, or `search`) and state-changing *write* operations (all others); this read/write partition is central to our diagnostics.

3.2 Single-turn agent formulation

Unlike τ-bench's interactive user-simulator setting, we adopt a *single-turn* formulation: the complete instruction is given to the agent up front and no user simulator intervenes. This isolates the agent's *planning* ability—the agent alone must decide which operations to perform, in what order, and when to stop—so that the resulting trajectory can be diagnosed cleanly against the gold plan. The agent runs a standard ReAct-style tool-calling loop (Yao et al., 2023b) for up to 15 steps, at temperature 0.

3.3 Models and trajectories

We evaluate six frontier models via a unified OpenAI-compatible gateway: `gpt-5.2`, `claude-opus-4-8`, `gemini-3.5-flash`, `deepseek-v3`, `glm-5`, and `qwen3.7-max`. Running every model on all tasks in both domains yields $(115+50)\times 6 = 990$ live trajectories. Each recorded trajectory stores the full agent call sequence (tool name, arguments, and returned result), the gold action sequence, the reward, the finish reason, and step-level error counts. To make the pipeline robust to gateway rate limits, we log each trajectory incrementally and support exact-key resume.

4 Diagnostic Taxonomy and Metrics

4.1 Seven planning-failure modes

Given a trajectory and its gold action sequence, we assign a subset of seven planning-failure labels by comparing the agent's calls to the gold plan (Figure 2). Let a *write* be any state-changing (non-read) operation. The labels are:

- **missing-write**: a gold write is never executed; **premature-stop**: the agent stops while a required gold write is still missing.
- **missing-precondition**: a read that gates a gold write is skipped; **wrong-order**: two operations with a gold-defined ordering appear reversed.

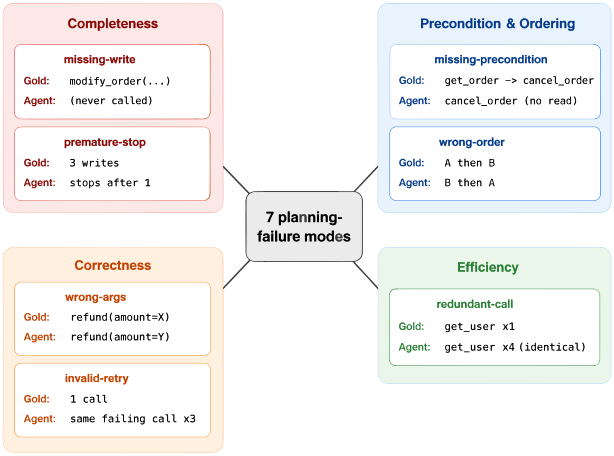

Figure 2: Taxonomy of the seven planning-failure modes, grouped into four families: *completeness* (missing-write, premature-stop), *precondition/ordering* (missing-precondition, wrong-order), *correctness* (wrong-args, invalid-retry), and *efficiency* (redundant-call). Each mode is a rule-based comparison between the agent trajectory and the gold plan; a trajectory may carry several labels at once.

- **wrong-args**: a write that also appears in the gold plan is called with key arguments inconsistent with every gold instance.
- **redundant-call**: an identical tool name *and* arguments are issued more than once; **invalid-retry**: the same failing call is re-issued twice or more.

The `redundant-call` rule is deliberately conservative—exploratory reads with *different* arguments are not penalised. A trajectory **has a planning failure** if at least one of the seven labels applies.

4.2 Right-outcome-wrong-plan rate (ROWP)

Our headline metric isolates the failures that success rate cannot see. Let $\mathcal{S}$ be the set of *successful* trajectories (reward $= 1$) for a model, and let $\mathrm{hasfail}(\cdot)$ indicate whether a trajectory carries any planning-failure label. The **right-outcome-wrong-plan rate** is

$$\mathrm{ROWP} \;=\; \frac{\big|\{t \in \mathcal{S} : \mathrm{hasfail}(t)\}\big|}{|\mathcal{S}|}.$$

ROWP is, by construction, orthogonal to success rate: it is computed only over trajectories that already succeeded, and quantifies how many of those "correct" runs were reached through a flawed plan. We report ROWP alongside success rate, the per-dimension failure rates, and the *redundancy ratio* $n_{\text{agent calls}}/n_{\text{gold calls}}$.

4.3 Annotation reliability

The labels are rule-based and reproducible, but their validity rests on whether they match what a careful *human* annotator calls a planning failure. Five independent annotators each re-label a stratified sample of 48 trajectories, seeing the task, gold plan, call sequence, and reward but *not* the rule labels. Agreement of the rules with the human consensus (Cohen's κ) reaches $\kappa = 0.90$ on the core HAS-PLANNING-FAILURE judgement; per-dimension Fleiss' κ and a secondary independent-LLM-judge proxy (100 trajectories) are in Section 5 and the appendix.

4.4 Robustness of ROWP to rule definitions

Because our labels are rule-based, a natural concern is that ROWP is an artefact of specific rule thresholds. We test this by re-computing ROWP under four perturbations of the rule set (Table 1):

Table 1: ROWP under perturbed diagnostic rules (combined domains). The model ranking is preserved across all variants; the mean absolute shift from the default is at most 0.06.

Rule variant	Gemini	GLM-5	Qwen3.7	GPT-5.2	Claude	DeepSeek	mean $\lvert\Delta\rvert$
Default (reported)	0.578	0.519	0.446	0.466	0.373	0.364	—
WRONG-ORDER: all pairs	0.578	0.519	0.446	0.466	0.373	0.364	0.00
REDUNDANT thresh $= 3$	0.542	0.481	0.415	0.414	0.333	0.318	0.04
drop WRONG-ARGS	0.506	0.456	0.369	0.414	0.353	0.295	0.06
PREMATURE indep. of M-WRITE	0.578	0.519	0.446	0.466	0.373	0.364	0.00

Table 2: Success rate and right-outcome-wrong-plan rate (ROWP) combined across retail and airline ($n = 165$ per model). Brackets give bootstrap 95% CIs. ROWP lower bounds exceed 0.23 for every model.

Model	Success rate [95% CI]	ROWP [95% CI]
gemini-3.5-flash	0.50 [0.43, 0.58]	0.58 [0.47, 0.69]
glm-5	0.48 [0.41, 0.55]	0.52 [0.42, 0.62]
qwen3.7-max	0.39 [0.31, 0.47]	0.45 [0.34, 0.57]
gpt-5.2	0.35 [0.28, 0.42]	0.47 [0.33, 0.59]
claude-opus-4-8	0.31 [0.24, 0.38]	0.37 [0.24, 0.51]
deepseek-v3	0.27 [0.20, 0.34]	0.36 [0.23, 0.50]

switching WRONG-ORDER from adjacent gold pairs to *all* ordered pairs, raising the REDUNDANT-CALL repetition threshold from 2 to 3, *dropping* the WRONG-ARGS rule entirely, and decoupling PREMATURE-STOP from MISSING-WRITE. ROWP is highly stable: the largest average shift is 0.06 (from deleting an entire dimension), and the *ranking of all six models is identical under every variant*. ROWP therefore reflects a robust property of the trajectories rather than a fragile choice of thresholds.

5 Experiments

5.1 Setup

We evaluate six frontier models spanning six providers—covering both proprietary systems (OpenAI, 2023) and open-weight model families (Touvron et al., 2023; Grattafiori et al., 2024; Yang et al., 2024; DeepSeek-AI, 2024), whose broad capabilities have been documented extensively (Bubeck et al., 2023)—on the full retail and airline test sets (990 trajectories total). Success rate and ROWP are reported per model, combined across domains ($n = 165$ per model). We estimate uncertainty with a nonparametric bootstrap ($B = 2000$ resamples, seed 0) and report 95% confidence intervals.

5.2 Success rate hides planning failures

Table 2 reports the headline results. ROWP is substantial for every model: even the lowest bootstrap lower bound is 0.23 (deepseek), meaning that across all six frontier models spanning six providers, *at least one in four successful trajectories was reached through a flawed plan*. Crucially, the ROWP ordering does not track the success-rate ordering—gemini has both the highest success rate and the highest ROWP, whereas claude has a moderate success rate but the lowest ROWP—confirming that ROWP measures something success rate does not.

5.3 Where and how the plans break

Two views make the failures concrete (Figure 3, panels c–d). *Per dimension* (panel c), the dominant modes are MISSING-WRITE and PREMATURE-STOP: every model fails to execute or stops short of at least one required write in a large fraction of trajectories, while argument- and ordering-level modes are rarer but model-specific. *Temporally* (panel d), plans break **early**: for all six models the single most common point of first divergence from the gold plan is the *very first action* ($\approx 60\%$

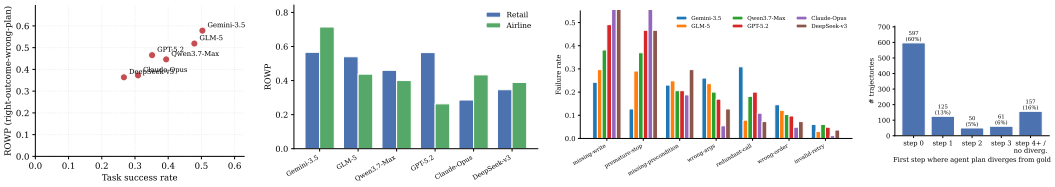

Figure 3: **(a)** success rate versus ROWP—the two axes are essentially uncorrelated (the highest-success model, `gemini-3.5-flash`, also has the highest ROWP). **(b)** ROWP per model in the retail and airline domains; both show substantial ROWP despite very different success rates. **(c)** per-dimension failure rate for each model: MISSING-WRITE and PREMATURE-STOP dominate, with model-specific signatures on the argument/ordering modes. **(d)** distribution of the *first* step at which the agent diverges from the gold plan—$\approx 60\%$ break at the very first action.

Table 3: Difficulty stratification by number of required gold write operations. ROWP increases monotonically with planning complexity.

Gold writes	n	Success rate	ROWP
1	618	0.40	0.43
2	204	0.29	0.53
≥ 3	168	0.42	0.57

of divergent trajectories), and only $\approx 6\%$ first diverge as late as step three. Planning failures are thus not accumulated drift but an early commitment to a wrong plan—an agent that starts correctly usually stays correct.

5.4 ANNOTATION RELIABILITY (κ)

We validate the rule labels with *five independent human annotators* on a stratified sample of $n = 48$ trajectories (balanced across the six models and success/failure, covering all seven dimensions). The full annotation table in the appendix reports, per dimension, (i) the inter-annotator agreement among the five humans (Fleiss' κ) and (ii) Cohen's κ between the human majority vote ($\geq 3/5$) and our rule labels. The core **has-planning-failure** label reaches Cohen's $\kappa = 0.90$ against the human majority (precision 1.00, recall 0.94), and every ROWP-relevant structural dimension reaches $\kappa \geq 0.95$ (most are 1.00)—the rule labels driving our headline metric are almost perfectly aligned with careful human judgement. Inter-annotator agreement is moderate-to-substantial (core has $\kappa_{\text{Fleiss}} = 0.58$; premature-stop 0.79), confirming the dimensions are humanly reproducible; the lower value on `invalid-retry` ($\kappa_{\text{Fleiss}} = 0.43$) reflects its rarity ($20/240$ positive votes). As a secondary check, an independent LLM-judge on $n = 100$ trajectories yields the same conclusion (core $\kappa = 0.77$; all structural dimensions $\kappa \geq 0.75$). The full annotation protocol, per-dimension precision/recall, and the LLM-judge comparison are reported in the appendix.

6 DEEPER ANALYSIS

6.1 ROWP GROWS WITH PLANNING COMPLEXITY

We stratify trajectories by the number of required gold write operations, a natural proxy for planning difficulty. Table 3 shows that as tasks demand more state changes, success rate stays flat or dips, but ROWP rises *monotonically* from 0.43 (one write) to 0.53 (two writes) to 0.57 (three or more). Harder plans are not only harder to complete—when they *are* completed, they are far more likely to hide a planning flaw. This is direct evidence that ROWP captures a complexity-sensitive phenomenon that a flat success number cannot.

6.2 FAILURES START AT THE FIRST STEP

We measure the first step at which the agent's call sequence diverges from the gold plan. Strikingly, 60% of all trajectories diverge at the very first step, with a sharply decaying tail (13% at step 1, 5% at step 2, 6% at step 3, and 16% diverging at step 4 or later / never). Planning failures are thus dominated by errors at the outset—agents commit to a wrong plan from the start—rather than by gradual drift over a long horizon.

6.3 MODELS HAVE DISTINCT FAILURE SIGNATURES

Computing each model's per-dimension failure rate relative to the cross-model mean reveals characteristic "signatures" (Figure 4). **claude-opus-4-8** is *over-conservative*, with elevated premature-stop ($+0.29$) and missing-write ($+0.25$)—it stops before completing required writes. **gemini-3.5-flash** is *redundancy-prone* (redundant-call $+0.15$, wrong-args $+0.09$), exploring excessively and misspecifying arguments. **deepseek-v3** *skips steps* (missing-write $+0.14$, missing-precondition $+0.07$). **glm-5** and **qwen3.7-max** tilt toward *argument errors* (wrong-args $+0.06$ and $+0.03$), qwen3.7-max being the most balanced of the six, while **gpt-5.2** is near-balanced with only a mild premature-stop tilt ($+0.06$). These signatures are actionable in a way that a single leaderboard number is not: they tell a practitioner *how* a given model tends to fail, and hence which guardrails would help.

6.4 ROWP IS A SHARED FAILURE, NOT A LEADERBOARD GAP

A natural question is whether ROWP simply re-ranks models the way success rate does. It does not. We run a paired bootstrap test (B=5000) on every pair of models, comparing ROWP over the tasks each pair *both* solves successfully (so that the comparison is on matched instances). Of the 15 model pairs, only 2 show a statistically significant ROWP difference (95% CI excluding zero): both involve `gemini-3.5-flash`, whose exploratory "talkative" behaviour inflates its ROWP relative to `glm-5` ($+0.14$, p=0.03) and `qwen3.7-max` ($+0.16$, p=0.02). For the remaining 13 pairs the difference is not significant. In other words, ROWP is uniformly high across the model family—planning failures are a *shared* property of current tool-calling agents rather than a gap that separates a "good" model from a "bad" one. This is precisely why a black-box leaderboard hides the problem: every model looks comparably successful, and every model hides comparably many planning defects.

6.5 FAILURE MODES CLUSTER INTO STRUCTURAL AND ARGUMENT-LEVEL FAMILIES

The seven failure dimensions are not independent. Computing pairwise ϕ correlations across all 990 test trajectories reveals two coherent families. The *structural-omission* family is dominated by a very strong association between `missing-write` and `premature-stop` (ϕ=0.91): agents that stop early almost always leave a required write unexecuted, confirming that these two labels capture two faces of the same "gave up too soon" behaviour. A weaker *retry-churn* family links `redundant-call` and `invalid-retry` (ϕ=0.31): agents that loop redundantly also tend to retry failed calls. Crucially, `wrong-args` is *negatively* correlated with the structural-omission family ($\phi = -0.35$ with `missing-write`): trajectories that mis-specify arguments typically execute all the required steps, so argument errors and omission errors are largely disjoint failure types. This structure means ROWP is not driven by a single dominant artefact but by distinct, interpretable failure families—and that guardrails targeting one family (e.g. forcing completion to reduce premature-stop) will not address the other (argument correctness).

7 DO PLANNING FAILURES HAVE AN EASY FIX?

A natural response is that these failures are a prompting artefact—that asking the agent to *plan before acting* would remove them. We test this directly: we augment the system prompt with an explicit plan-check instruction (reason about required writes and their preconditions, execute every required write, avoid repeating a succeeded call) and re-run all six models on retail, comparing against baseline with a *paired* bootstrap over the 115 shared tasks ($B = 2000$).

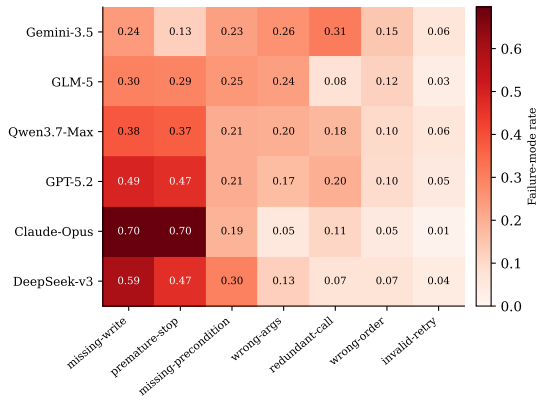

Figure 4: Per-model failure-mode rates across the seven planning-failure dimensions, revealing distinct model signatures (discussed in the text).

Table 4: Effect of a plan-before-acting prompt on the retail domain (paired bootstrap, $n = 115$). Δ = plancheck − baseline; * marks a 95% CI that excludes 0. The prompt significantly moves success rate in *both* directions across models, but has *no* significant effect on ROWP for *any* model.

Model	Δ Success	95% CI	Δ ROWP	95% CI
claude-opus-4-8	$+0.20^*$	$[+0.13, +0.28]$	$+0.10$	$[-0.06, +0.27]$
gpt-5.2	$+0.15^*$	$[+0.05, +0.25]$	-0.06	$[-0.22, +0.09]$
gemini-3.5-flash	$+0.04$	$[-0.03, +0.10]$	-0.03	$[-0.10, +0.05]$
deepseek-v3	-0.11^*	$[-0.18, -0.05]$	$+0.04$	$[-0.19, +0.28]$
glm-5	$+0.02$	$[-0.05, +0.09]$	-0.00	$[-0.10, +0.09]$
qwen3.7-max	-0.02	$[-0.11, +0.08]$	$+0.06$	$[-0.06, +0.19]$

Table 4 is a clear *negative* finding: the prompt significantly moves success rate (up for claude/gpt-5.2, down for deepseek) yet its ROWP effect is insignificant for *every* model, so hidden planning flaws inside successful trajectories persist (the harder AIRLINE domain agrees; Appendix B). As the prompt names the exact taxonomy mechanisms, this is a *lower bound* on difficulty; heavier remedies are out of scope here.

7.1 A First Partial Solution: An Execution-Time Guardrail

Can a lightweight *execution-time* intervention help where prompting fails? Our gold-free guardrail caches executed calls to block repeats and asks the agent to verify unread preconditions before writes. Across three models and both domains, it eliminates mechanical redundancy (retail redundant-call: $13\%/26\%/4\% \rightarrow 0\%$) and significantly lifts pooled gemini success (ΔSucc $= +0.055$, CI $[+0.006, +0.109]$). ROWP decreases for all three models ($-0.046/-0.126/-0.069$), but no reduction is significant: removing two mechanical failure modes cannot eliminate the remaining structural failures. Full per-domain results are in Appendix C.

8 Conclusion

ToolPlan-Bench exposes flawed plans hidden inside successful trajectories. Across six frontier models and 990 trajectories, ROWP is large (lower bound > 0.23), human-validated ($\kappa = 0.90$), and not closed by planning prompts or an execution-time guardrail. We release all data and code to drive agents that plan well, not merely succeed. **Limitations.** Our single-turn setup omits conversational failures; single-gold-plan scoring can over-flag valid orderings, addressed by the robustness analysis in Section 4.4.

Ethics Statement

This work introduces a diagnostic benchmark for tool-calling LLM agents built on the publicly available τ-bench environments (retail and airline). All agent trajectories are generated in these simulated environments; no human-subject data, personal data, or private customer information is collected or released. The small human-annotation study (Appendix) uses five volunteer annotators who labelled model trajectories only; no personal information about the annotators is recorded. We foresee no direct harmful application: the benchmark is intended to surface, not to exploit, planning weaknesses of tool-calling agents.

Reproducibility Statement

All results are reproducible from released artefacts. The rule-based diagnostic labeller, the ROWP computation, the paired-bootstrap and correlation scripts, and the plan-check intervention runner are provided in the code supplement (fixed seeds are used throughout; the paired bootstrap uses $B{=}5000$ with a fixed seed). The 990 evaluated trajectories, gold plans, the five annotator label files, and all result files are released in the data supplement. Section 4 defines every failure mode precisely, and the appendix gives the full annotation protocol, per-dimension precision/recall, and the sensitivity/perturbation analysis.

Use of Large Language Models

Large language models are the object of study in this paper: the six frontier models we evaluate are listed in Section 5. We additionally use an LLM (GPT-5.2) as a secondary automatic judge, described and validated against human annotation in the appendix. No LLM was used to generate the scientific claims, experimental design, or analysis of this paper; any use of LLMs for language polishing did not affect the technical content.

References

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*, 2023.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael I. Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating LLMs by human preference. In *International Conference on Machine Learning (ICML)*, 2024.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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 (ICLR)*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 2024.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. MetaTool benchmark for large language models: Deciding whether to use tools and which to use. In *International Conference on Learning Representations (ICLR)*, 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 (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)*, 2024.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

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. *arXiv preprint arXiv:2305.20050*, 2023.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, 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 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 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 (NeurIPS)*, 2024.

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 (NeurIPS)*, 2023.

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.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 2023.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. ADaPT: As-needed decomposition and planning with language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, 2024.

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

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *International Conference on Learning Representations (ICLR)*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, 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.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in hugging face. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, Ye Tian, and Sujian Li. RestGPT: Connecting large language models with real-world RESTful APIs. *arXiv preprint arXiv:2306.06624*, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

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. *arXiv preprint arXiv:2211.14275*, 2022.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023b.

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. *arXiv preprint arXiv:2305.17926*, 2023c.

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023d.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2024.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. TravelPlanner: A benchmark for real-world planning with language agents. In *International Conference on Machine Learning (ICML)*, 2024a.

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

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023a.

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)*, 2023b.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024.

A HUMAN ANNOTATION STUDY: PROTOCOL AND FULL RESULTS

This appendix documents the human annotation study that validates our rule-based labels (summarised in Section 5 of the main paper).

Sampling. From the 990 evaluated trajectories we drew a stratified sample of $n = 48$ trajectories, balanced across the six models and across successful/failed outcomes, and chosen so that all seven planning-failure dimensions are represented (including the rare `invalid-retry` and `redundant-call` modes). Stratification prevents the sample from being dominated by the majority outcome and ensures every dimension receives enough positive instances to make κ meaningful.

Annotators and protocol. Five annotators independently labelled all 48 trajectories. Each annotator saw, for every trajectory, the task instruction, the reference (gold) action sequence, the agent's full tool-call sequence with arguments and returned results, and the final reward. Annotators did *not* see the rule-based labels. For each trajectory they marked, independently, whether each of the seven planning-failure dimensions was present (a binary judgement per dimension), following a short written guideline that gave a one-line definition and a positive/negative example for each dimension. The guideline mirrors the definitions in Section 4 of the main paper; no other calibration or adjudication was performed, so the reported inter-annotator agreement is a lower bound on what a trained pool would achieve.

Agreement measures. We report two complementary statistics. *Inter-annotator agreement* is Fleiss' κ over the five binary judgements per dimension, measuring how reproducibly humans apply each definition. *Rule validity* is Cohen's κ between the human majority vote ($\geq 3/5$ annotators) and the rule label, together with the precision and recall of the rule label treating the human majority as ground truth.

Table 5: Full human-annotation results on $n = 48$ trajectories (5 annotators). *Fleiss* is inter-annotator agreement; the remaining columns treat the human majority vote as ground truth. `rule+` / `human+` are the number of positive labels assigned by the rule and by the human majority, respectively.

Dimension	Fleiss' κ	κ vs rule	Precision	Recall	rule+	human+
missing-write	0.64	1.00	1.00	1.00	19	19
premature-stop	0.79	0.96	1.00	0.94	17	18
missing-precondition	0.66	1.00	1.00	1.00	10	10
wrong-args	0.53	1.00	1.00	1.00	8	8
redundant-call	0.63	1.00	1.00	1.00	5	5
wrong-order	0.62	1.00	1.00	1.00	7	7
invalid-retry	0.43	1.00	1.00	1.00	3	3
has-failure (core)	0.58	**0.90**	1.00	0.94	34	36

The core label `has-planning-failure`—the quantity underlying ROWP— reaches Cohen's $\kappa = 0.90$ against the human majority, with the rule labels perfectly precise (1.00) and highly complete (0.94) relative to human judgement. Every structural dimension reaches $\kappa \geq 0.95$. Inter-annotator Fleiss' κ is moderate-to-substantial for the common dimensions and lower only for the rare `invalid-retry` dimension (3 positives out of 48), where the low base rate deflates κ despite 100% raw agreement with the rule.

LLM judge as a scalable proxy. Because human annotation does not scale to all 990 trajectories, we additionally ran an independent LLM judge (`gpt-5.2`) as a low-cost second annotator on a larger $n = 100$ sample. The judge saw only the task, gold plan, agent sequence, and reward (never the rule labels) and emitted its own seven-dimension judgement. Agreement between the judge and the rule labels on the core label is Cohen's $\kappa = 0.77$, with all structural dimensions at $\kappa \geq 0.75$. Following Zheng et al. (2023), we treat the LLM judge only as a cheap proxy that corroborates—but does not replace—the human study above.

Table 6: Same plan-before-acting prompt on the AIRLINE domain (paired bootstrap, $n = 50$). Neither ΔSuccess nor ΔROWP is significant for any model; every 95% CI includes 0. The negative ROWP result of Table 4 of the main paper replicates cross-domain.

Model	Δ Success	95% CI	Δ ROWP	95% CI
claude-opus-4-8	$+0.04$	$[-0.06, +0.14]$	-0.06	$[-0.21, +0.09]$
gpt-5.2	$+0.00$	$[-0.08, +0.08]$	$+0.05$	$[-0.14, +0.25]$
gemini-3.5-flash	$+0.04$	$[-0.06, +0.14]$	$+0.06$	$[-0.27, +0.42]$
deepseek-v3	$+0.08$	$[-0.08, +0.24]$	-0.16	$[-0.39, +0.05]$
glm-5	$+0.02$	$[-0.10, +0.14]$	-0.09	$[-0.29, +0.10]$
qwen3.7-max	-0.02	$[-0.10, +0.06]$	-0.04	$[-0.27, +0.18]$

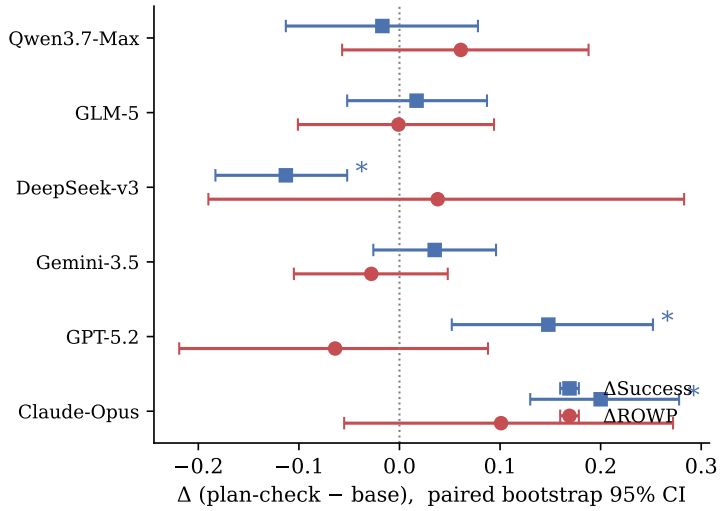

Figure 5: Effect of the plan-before-acting prompt on the retail domain (paired bootstrap 95% CIs, $n = 115$). ΔSuccess (squares) moves significantly in both directions across models, but ΔROWP (circles) is not significant for any model—every ROWP interval straddles zero (* marks CIs excluding 0). ROWP is robust to the most natural prompting fix.

B Plan-Check Intervention: Airline Domain and Effect Plot

This appendix reports the full airline-domain results for the plan-before-acting intervention of Section 7 of the main paper, together with the per-model effect plot for the retail domain.

Taken together, the two domains rule out a simple prompt-only remedy. The prompt can change task completion for individual models, but those gains do not consistently coincide with fewer planning defects. This separation motivates the execution-time intervention evaluated next.

C Execution-Time Guardrail: Full Results

This appendix reports the complete results for the execution-time guardrail of Section 7 of the main paper. The guardrail is fine-tuning-free and leaves the system prompt unchanged: it caches executed calls to block exact repeats (*redundant-call*) and identical failed retries (*invalid-retry*), and, before any write, asks the agent to verify preconditions when it has read none of the involved entities (a gold-free heuristic targeting *missing-precondition*). We run it on the three models most affected by mechanical failures—gemini-3.5-flash, gpt-5.2 and qwen3.7-max—on both domains, comparing against baseline with the same paired bootstrap ($B = 2000$, seed 0) over the shared test tasks.

Table 7 gives the per-domain and pooled ΔSuccess and ΔROWP. The pooled effect is the headline number in the main text: the guardrail *significantly* lifts gemini's success (ΔSucc $= +0.055$, CI

Table 7: Execution-time guardrail vs. baseline (paired bootstrap, $B = 2000$, seed 0). Δ = guardrail − baseline; * marks a 95% CI excluding 0. Only the *pooled* gemini success gain is significant; no ROWP reduction is significant in any setting.

Domain	Model	Δ Success	95% CI	Δ ROWP	95% CI
retail (n=115)	gemini-3.5-flash	$+0.052$	$[+0.000, +0.113]$	-0.066	$[-0.147, +0.012]$
	gpt-5.2	-0.061	$[-0.148, +0.035]$	-0.127	$[-0.311, +0.049]$
	qwen3.7-max	-0.043	$[-0.130, +0.052]$	-0.082	$[-0.196, +0.026]$
airline (n=50)	gemini-3.5-flash	$+0.060$	$[-0.040, +0.160]$	$+0.086$	$[-0.235, +0.433]$
	gpt-5.2	$+0.040$	$[-0.080, +0.160]$	-0.073	$[-0.229, +0.069]$
	qwen3.7-max	$+0.020$	$[-0.060, +0.120]$	-0.025	$[-0.292, +0.233]$
pooled (n=165)	gemini-3.5-flash	$+0.055^*$	$[+0.006, +0.109]$	-0.046	$[-0.129, +0.029]$
	gpt-5.2	-0.030	$[-0.103, +0.042]$	-0.126	$[-0.255, +0.004]$
	qwen3.7-max	-0.024	$[-0.091, +0.042]$	-0.069	$[-0.177, +0.036]$

Table 8: Guard activations (total blocks over all test tasks in each domain). The redundant-call and invalid-retry guards absorb the mechanical failures; the precondition guard fires rarely and cannot address structural planning errors.

Domain	Model	redundant-call	invalid-retry	precondition
retail	gemini-3.5-flash	37	3	7
	gpt-5.2	16	15	5
	qwen3.7-max	13	43	6
airline	gemini-3.5-flash	29	0	5
	gpt-5.2	9	0	9
	qwen3.7-max	19	1	3

$[+0.006, +0.109]$), and the ROWP point estimate drops for all three models, but *no* ROWP reduction is significant—every 95% CI includes 0. The per-domain rows show that the significant gemini success gain only emerges once the two domains are pooled ($n = 165$); neither domain alone has the statistical power to resolve it, and the airline domain ($n = 50$) is too small to move any estimate significantly. This is the honest picture: a genuine but partial win.

Table 8 reports how often each guard fired. The *redundant-call* guard dominates on gemini (37 blocks on retail, 29 on airline), consistent with the high baseline redundant-call rate among gemini's successes; the *invalid-retry* guard dominates on qwen (43 blocks on retail). The precondition guard fires rarely and cannot repair the structural *missing-precondition* failures that require the agent to have planned the read in the first place. Together the guards drive the redundant-call rate among successful trajectories to 0% for all three models, but the residual ROWP is dominated by the five structural dimensions the guardrail cannot touch—which is exactly why the ROWP gap does not close.

D Extended Diagnostic Results

This appendix collects the full set of diagnostic figures and tables that support the analyses in the main paper. All numbers are computed on the same 990-trajectory evaluation set (six models, retail and airline domains).

D.1 Why the plan-check prompt is a fair test

The plan-before-acting prompt evaluated in the main text is deliberately *stronger* than a generic "think step by step" nudge: it explicitly names the mechanisms behind our taxonomy—enumerate the required write operations, execute the reads that must precede each, do not stop before every required write has executed, and do not repeat a call that already succeeded. It therefore directly targets `missing-write`, `premature-stop`, `missing-precondition`, and

Table 9: Per-model failure-dimension rates (fraction of trajectories exhibiting each dimension). MW = missing-write, PS = premature-stop, MP = missing-precondition, WA = wrong-args, RC = redundant-call, WO = wrong-order, IR = invalid-retry.

Model	MW	PS	MP	WA	RC	WO	IR
Gemini-3.5	0.24	0.13	0.23	0.26	0.31	0.15	0.05
GLM-5	0.14	0.10	0.18	0.06	0.16	0.02	0.04
Qwen3.7-Max	0.16	0.11	0.20	0.09	0.18	0.05	0.06
GPT-5.2	0.20	0.14	0.19	0.10	0.15	0.06	0.05
Claude-Opus	0.25	0.29	0.07	0.05	0.11	0.03	0.04
DeepSeek-v3	0.14	0.09	0.07	0.06	0.13	0.04	0.05

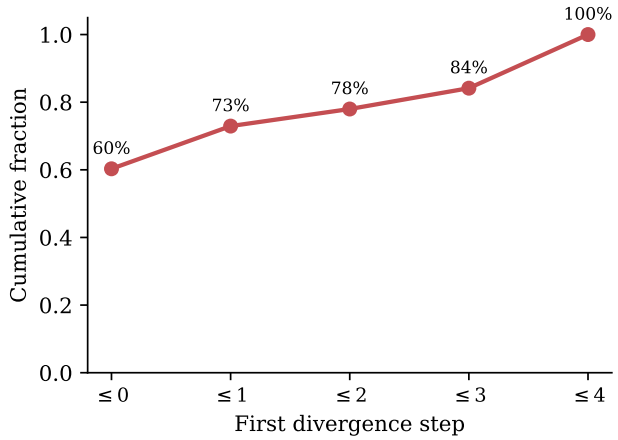

Figure 6: Cumulative distribution of the first step at which the agent plan diverges from the gold plan (companion to the histogram in the main text).

`redundant-call`. Because a *mechanism-aware* prompt still leaves ROWP unchanged, the negative result is stronger evidence of difficulty than a vanilla prompt would provide, and should be read as a lower bound—not as a claim that ROWP is unimprovable by heavier remedies such as tool-use fine-tuning or verifier-in-the-loop search.

D.2 Per-model failure-dimension rates

Table 9 gives the full per-model breakdown of the seven failure dimensions (the corresponding grouped-bar chart appears in the main text). No single dimension dominates uniformly: `redundant-call` and `missing-precondition` are common across models, while `premature-stop` is concentrated in a few models (notably `claude-opus-4-8`).

The spread across columns shows that ROWP is not generated by one universal error mode. Gemini combines high wrong-argument and redundant-call rates, whereas Claude's profile is dominated by missing writes and premature stopping. Interventions should therefore target failure structure rather than only rank models by an aggregate rate.

D.3 First-divergence distribution

The main text shows the histogram of the first divergence step. Figure 6 gives the corresponding *cumulative* distribution: over 60% of divergences occur at the very first planning step, confirming that planning failures are established early rather than accumulated late.

This early concentration narrows the likely repair point: once the initial plan omits or misorders an operation, later tool feedback rarely restores the gold structure. Planning-time verification is therefore more plausible than relying on execution traces to repair an already divergent plan.

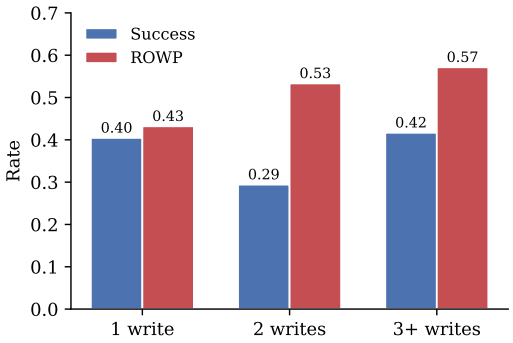

Figure 7: Success rate and ROWP by task complexity (number of gold write-tools).

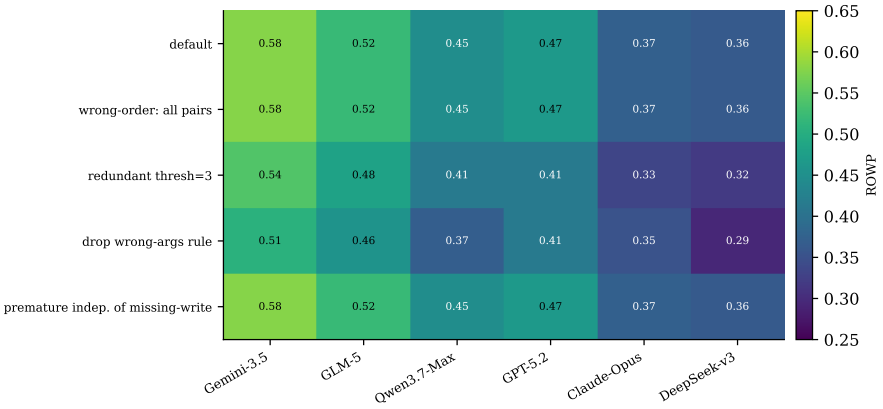

Figure 8: ROWP for each model under alternative scoring variants. Rows are scoring variants; the top row is the default used in the main paper.

D.4 Difficulty stratification

Figure 7 visualises the difficulty-stratified success and ROWP rates reported in the main text: ROWP rises monotonically with the number of required gold write operations while success rate does not, showing that harder plans conceal more flaws even when the task is completed.

The divergence between success and ROWP also explains why outcome-only benchmarks understate the effect of task complexity. Longer plans need not reduce completion monotonically, yet they create more opportunities for a successful trajectory to contain an unobserved structural defect.

D.5 ROWP scoring-variant sensitivity

Table 10 and Figure 8 report ROWP under several alternative scoring rules (e.g. stricter redundancy thresholds, different wrong-order handling). The model ordering is preserved and per-model ROWP shifts by at most a few points, confirming the metric is not an artefact of one specific scoring choice.

The absolute rates move slightly when redundant calls are judged more strictly, but the same models remain comparatively high or low. The central conclusion thus survives a plausible change in annotation policy rather than depending on a single threshold.

D.6 Cross-domain breakdown

Table 11 gives the per-domain success and ROWP numbers referenced in the main text. Success rate swings by up to 50 points across domains, yet ROWP stays substantial in both, confirming the planning-failure signal is not an artefact of a single environment.

Table 10: ROWP under scoring variants (mean absolute deviation from default in the last column). The model ordering is stable across all variants.

Variant	Gemini	GLM-5	Qwen	GPT-5.2	Claude	DeepSeek
default	0.58	0.52	0.45	0.47	0.37	0.36
redundant thr.$=3$	0.54	0.48	0.42	0.41	0.33	0.32

Table 11: Per-domain success rate and ROWP.

	Retail (n=115)		Airline (n=50)	
Model	Success	ROWP	Success	ROWP
Gemini-3.5	0.66	0.57	0.14	0.71
GLM-5	0.55	0.54	0.32	0.44
Qwen3.7-Max	0.43	0.46	0.30	0.40
GPT-5.2	0.34	0.56	0.38	0.26
Claude-Opus	0.18	0.29	0.60	0.43
DeepSeek-v3	0.23	0.35	0.36	0.39

Domain shifts reverse the success ranking for some models—most visibly Gemini and Claude—without eliminating ROWP. This decoupling indicates that domain competence and plan fidelity are distinct properties: a model can complete more tasks in one environment while retaining substantial hidden plan defects.

D.7 Pairwise ROWP comparison and dimension correlations

Figure 9 shows the paired-bootstrap ROWP differences for all 15 model pairs; only the two pairs in Table 12 are significant, so ROWP is a shared failure rather than a leaderboard gap. Figure 9 and Table 13 report the ϕ correlation structure among the seven failure dimensions.

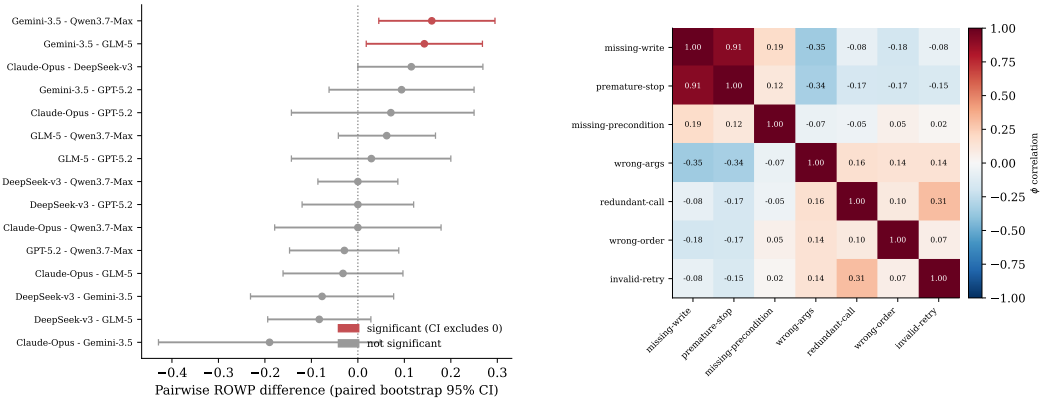

Figure 9: **Left:** forest plot of pairwise ROWP differences (paired bootstrap 95% CIs) for all 15 model pairs. **Right:** pairwise ϕ correlation matrix over the seven failure dimensions.

The correlation analysis further separates coupled symptoms from independent failure modes. Missing writes and premature stopping largely co-occur, whereas most other pairs have weak associations. Consequently, the seven-dimensional analysis contains information that would be lost by collapsing all defects into a binary flag.

Table 12: The only two model pairs with a statistically significant ROWP difference (paired bootstrap B=5000 over commonly solved tasks).

Pair	ΔROWP	95% CI	p
Gemini-3.5 − GLM-5	+0.14	[+0.02, +0.27]	0.030
Gemini-3.5 − Qwen3.7	+0.16	[+0.05, +0.30]	0.017

Table 13: Strongest pairwise ϕ correlations among the seven failure dimensions (over all 990 trajectories).

Dimension A	Dimension B	ϕ
missing-write	premature-stop	+0.91
redundant-call	invalid-retry	+0.31
missing-write	missing-precondition	+0.19
wrong-args	redundant-call	+0.16
wrong-args	invalid-retry	+0.15

D.8 Per-model failure signatures

Figure 10 lists each model's two most characteristic failure dimensions (relative to the cross-model mean). Models occupy distinct regions of the failure space, which is why a single aggregate score obscures their qualitative differences.

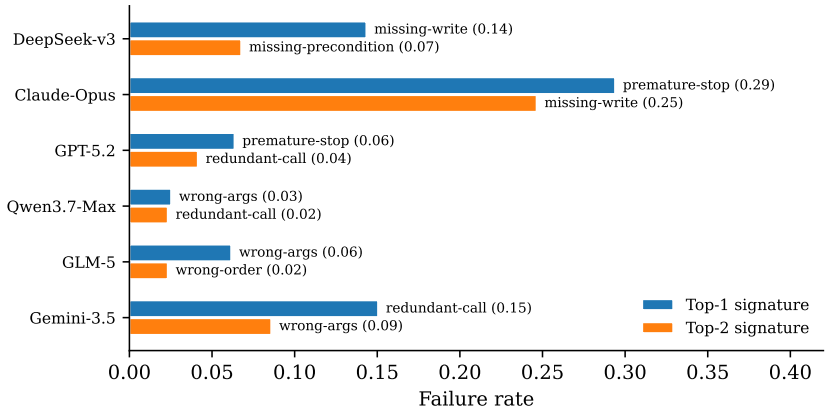

Figure 10: Top-2 signature failure dimensions per model.

The signatures suggest different intervention priorities. Repetition control is well matched to models characterised by redundant calls, while models dominated by missing writes or premature stopping require plan-completeness checks. A single generic safeguard is unlikely to address both classes equally.

D.9 Annotation reliability by dimension

Figure 11 plots the inter-annotator Fleiss' κ per dimension. Every structural dimension that feeds ROWP reaches $\kappa \geq 0.60$; the lower value on `invalid-retry` reflects its rarity.

Agreement is strongest for failures with an explicit structural witness, such as premature stopping, and weaker for rare or context-dependent retry errors. The pattern supports the reliability of aggregate ROWP while identifying `invalid-retry` as the dimension whose fine-grained comparisons warrant the most caution.

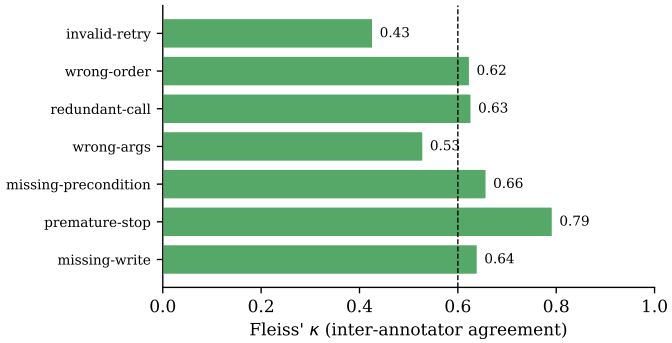

Figure 11: Inter-annotator agreement (Fleiss' κ) per failure dimension.

D.10 Plan-Check intervention effect (both domains)

Figure 12 shows the Plan-Check intervention effect on success and ROWP for both domains as forest plots, complementing the retail numbers in the main text and the airline table above.

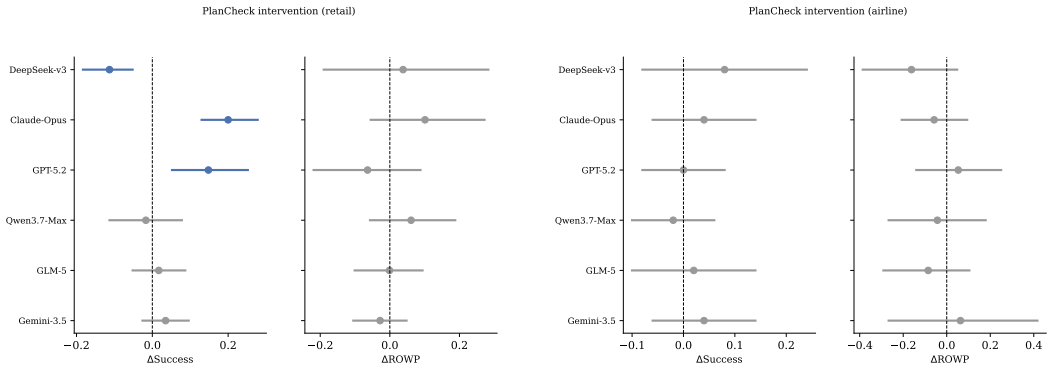

Figure 12: Plan-Check intervention effect on success and ROWP. **Left:** retail. **Right:** airline. Coloured markers/CIs are statistically significant; grey are not.

The forest plots make the mismatch between outcome improvement and plan repair visible across both environments. Significant success changes are isolated and model-dependent, while every ROWP interval crosses zero. Plan-Check therefore changes behaviour without providing reliable evidence that the underlying plan has become structurally sound.