TESTING AGENT PROCESS VERIFIERS WITH COUNTERFACTUAL INSTRUCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Process verifiers guide tool-using agents by ranking candidate actions. However, evaluating a verifier on a single instruction cannot tell whether its preference actually depends on the task requirement. We introduce crossed instruction–trajectory matching: the same two candidate trajectories are evaluated under two instructions that differ in one requirement. A separate action-level predicate recomputes the required argument from the instruction and shared history, and verifies which candidate action is correct. A verifier must reverse its preference to solve the group, ruling out a fixed candidate preference. Using this design, we construct 195 groups from three tool-use benchmarks and evaluate five prompted judges and five trained reward models. On selection-by-attribute groups, 12 of 42 judge–group pairs answered correctly under the original instruction fail under the edited instruction, including seven fixed candidate preferences. Judges with similar crossed accuracy fail in different ways, including fixed preferences, position-following, and ties. Organizing the listed options by item improves all five judges on both diagnosed and held-out groups, whereas simply restating the criterion does not consistently help. Raw-value tables, which group observed values by item without computing attributes, recover 5–10 of 11 held-out groups, and computed tables recover 8–11. Because success requires correct choices under both instructions, this recovery cannot be explained by a stronger fixed preference for one candidate. Finally, we show that evaluation protocol matters: scoring candidates separately can lose distinctions recovered by direct comparison, and aggregating step-level scores by their minimum can erase correct local error detection.

1 INTRODUCTION

Agents interact with the world through sequences of tool calls (Yang et al., 2024). Process verifiers score intermediate actions or trajectory prefixes, providing denser feedback than final task success alone. These scores can guide online trajectory search or reinforcement learning (RL) (Lightman et al., 2023; Wang et al., 2024b). Incorrect verifier feedback can favor unsuccessful trajectories or reinforce wrong actions. Reliable evaluation therefore requires checking whether a verifier's preference actually depends on the task requirement.

Existing benchmarks typically measure agreement with correctness labels on actions or complete trajectories collected from human or model rollouts (Lambert et al., 2025). On natural rollouts, execution errors, repetition, or trajectory length can correlate with failure (Wang et al., 2026a). Agreement with correctness labels therefore does not establish that a verifier distinguishes actions that satisfy the instruction from those that do not, rather than relying on these correlated cues. Figure 1(a) illustrates the distinction: moving a file into `temp` can execute successfully as a tool call even though the instruction requests `archive`. Execution success alone therefore does not establish that the action satisfies the instruction. Even a correct choice is ambiguous when only one instruction is tested: a verifier may prefer the same candidate regardless of the requested destination.

We introduce crossed instruction–trajectory matching to test whether a verifier's preference follows a changed task requirement. Each group evaluates the same two candidate trajectories under two instructions that differ in one requirement. The trajectories share a prefix and diverge at one action, so the verifier must change its preference when the required argument changes. Neither trajectory raises a tool-execution error, and execution against the benchmark establishes that each trajectory

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.

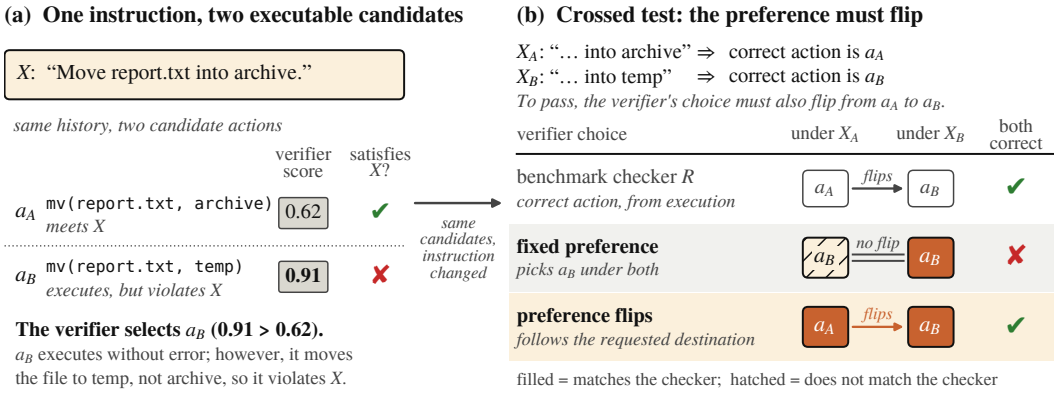

Figure 1: **(a)** A verifier can prefer an executable but instruction-violating action. **(b)** In a crossed group, the same two candidates are scored under two instructions that differ in one requirement. The correct candidate changes from a_A under X_A to a_B under X_B, so a verifier passes only if its preference changes with the instruction. A fixed preference is correct in at most one row.

succeeds under exactly one instruction. In Figure 1(b), the verifier must prefer moving the file into `archive` under one instruction and into `temp` under the other. An independently implemented action-level predicate checks, from the instruction and prior observations, which candidate action satisfies the edited requirement. These checks verify the labels for all four instruction–trajectory combinations, rather than assuming that changing the instruction changes correctness. Requiring the verifier to switch to the candidate that matches the changed requirement rules out a fixed preference for either candidate. In our primary evaluation, the verifier sees the instruction, the shared history, and the candidate actions, but not their resulting observations or any later steps. It must therefore judge the decision before its consequences are revealed.

We construct 195 execution-verified groups from BFCL multi-turn (Patil et al., 2025), ToolSandbox (Lu et al., 2024), and τ^2-bench (Barres et al., 2026), and evaluate five prompted judges and five trained reward models. We organize groups by how the target argument is determined: stated values, fixed mappings, observation lookups, selection by attribute, and observation-based computations. This separation matters because some groups can be solved by surface matching, whereas others require information from prior observations. Our main findings are:

- **Testing both instructions exposes failures hidden by single-instruction accuracy.** On selection-by-attribute groups, 12 of 42 judge–group pairs answered correctly under the original instruction fail under the edited instruction. Judges with similar crossed accuracy fail through fixed preferences, position-following, or ties (§4.2).

- **Selection improves when item attributes are organized for comparison.** Mechanical flattening helps simple listings but remains weak on multi-leg itineraries. Raw-value tables recover 5–10 of 11 held-out groups without computing attributes, and computed tables recover 8–11 (§4.3).

- **Scoring and aggregation affect whether judgments support a correct ranking.** On controlled decisions, a judge can assign low scores to both candidate actions when each is presented in a separate evaluation, yet choose correctly when both are presented together. At the trajectory level, aggregating step scores by their minimum can erase correct local error detection when both trajectories contain a low-scored step. We compare scoring protocols and aggregation rules on the same groups to measure these effects (§4.4).

Together, these tests separate whether a verifier happens to choose the correct candidate from whether its ranking actually responds to the task requirement and remains informative under the scoring protocol.

2 Related Work

Process reward models and step-level judges. Process supervision scores intermediate steps instead of final answers and was first developed for mathematical reasoning (Lightman et al., 2023; Wang et al., 2024b; Zhang et al., 2025). For agents, step scores come from prompted judges that answer rubric questions about a trajectory (Lù et al., 2025; Chae et al., 2025) or from reward models trained on tool-use data (Choudhury, 2025; Li et al., 2025; Lin et al., 2025). We evaluate both families under a common protocol and treat the scoring format (pointwise step labels, pairwise comparison, prefix scoring) as a controlled variable rather than as a property of the model, because the format determines what the verifier is asked to compute.

Meta-evaluation of verifiers. ProcessBench (Zheng et al., 2024) and PRMBench (Song et al., 2025) measure whether process reward models identify erroneous steps in mathematical reasoning, while AgentRewardBench (Lù et al., 2025) and AgentProcessBench (Fan et al., 2026) evaluate reward or process judgments on agent trajectories. These benchmarks largely evaluate natural trajectories, where policy-generated errors can mix instruction inconsistency with other trajectory properties. Shen et al. (2024) test reward consistency across similar instructions and their corresponding responses, while Tiwari et al. (2026) perturb traces while holding the problem fixed. We instead construct agent decisions in which the same trajectories are evaluated under instructions differing in one requirement, share a prefix and diverge at one action, and are validated across all four combinations by execution and an independent action-level predicate. Indirect evaluation through policies trained against a verifier (Wang et al., 2026b; Zhang et al., 2026) further mixes verifier quality with the training procedure and visited states (Gao et al., 2023).

Metamorphic and counterfactual testing. Controlled input perturbations have long been used to test known relations between outputs in metamorphic testing (Chen et al., 1998). In NLP, contrast sets apply small, meaningful changes that typically alter the expected label (Gardner et al., 2020), while CheckList tests invariance and directional expectations under controlled transformations (Ribeiro et al., 2020). Metamorphic testing has also been applied directly to LLMs (Cho et al., 2025). Our setting differs in that an instruction edit does not determine the counterfactual label by construction. We execute all four instruction–trajectory combinations and retain a group only when the benchmark checker verifies the crossed pattern. Thus, the test measures whether the verifier's preference follows the changed requirement without assuming which trajectory the edit should make correct.

Evaluation protocols and judge biases. LLM judges can exhibit position bias in pairwise evaluation, motivating evaluation in both presentation orders (Wang et al., 2024a; Shi et al., 2025). More generally, pointwise and pairwise feedback can induce different evaluation behavior and biases (Tripathi et al., 2025). We compare these protocols on identical instructions, prefixes, and candidate actions, so that differences can be attributed to the scoring protocol rather than to different evaluation examples.

3 Crossed Instruction–Trajectory Matching

We construct decisions whose correct candidate changes with one instruction requirement, verify their trajectory- and action-level labels, and test whether a verifier reverses its preference. We then compare scoring protocols on the same groups.

3.1 The crossed test

We construct pairs of tasks in which changing one instruction requirement changes which of two candidate actions is correct. A task consists of an instruction X, an executable environment, and a ground-truth reward $R(T \mid X) \in \{0, 1\}$ for a trajectory T. A process verifier V scores a candidate action or trajectory given X. A crossed group tests whether the verifier changes its preference when the task requirement changes. The same two candidate trajectories, T_A and T_B, are evaluated under two instructions, X_A and X_B, that differ in one requirement. Both trajectories execute successfully, but the ground-truth reward is 1 for T_A only under X_A and for T_B only under X_B. We call (X_A, T_A) and (X_B, T_B) the matched cells, and the two crossed combinations the mismatched cells.

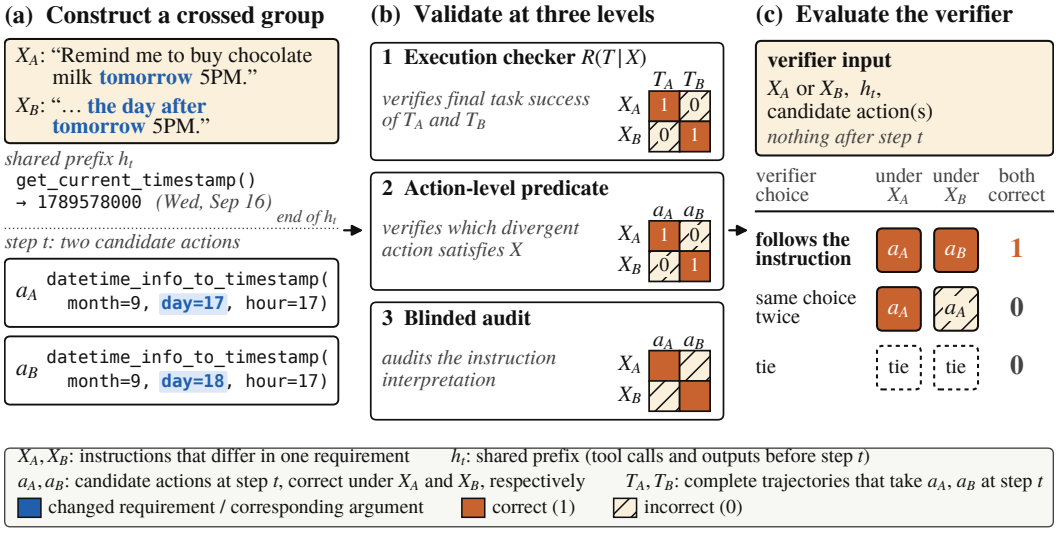

Figure 2: Construction, validation, and evaluation of a crossed group. **(a)** Editing one instruction requirement changes the corresponding argument, producing candidate actions a_A and a_B after a shared prefix h_t. **(b)** The complete trajectories are checked by the benchmark reward, while an independent action-level predicate verifies which divergent action satisfies each instruction; a blinded audit checks the instruction interpretation. **(c)** The verifier receives the instruction, shared prefix, and candidate actions, and must select the correct candidate under both instructions.

Construction requirements. For each crossed group, we enforce three requirements. A crossed group consists of two instructions X_A, X_B and two trajectories T_A, T_B that share a prefix h_t and take actions a_A, a_B at step t, such that:

(C1) *One requirement changes.* X_A and X_B differ in one requirement, and this requirement determines one argument of the call at step t. Every mention of the requirement is edited consistently, including repeated mentions and aliases. All other information in the instruction is kept unchanged. (Appendix E). The actions a_A and a_B are the same call and differ only in that argument. After step t, the trajectories may differ only in downstream arguments that depend on the edited value and in the resulting observations.

(C2) *The successful trajectory switches.* Every call in both trajectories executes without error, and the ground-truth reward is 1 exactly for the matched cells:

$$R(T_A \mid X_A) = R(T_B \mid X_B) = 1, \qquad R(T_B \mid X_A) = R(T_A \mid X_B) = 0. \tag{1}$$

(C3) *The divergent action is independently verified.* The shared prefix h_t contains every observation needed to determine the edited argument. No step of h_t depends on the edited requirement, so it is valid under both instructions. We use an independently implemented predicate $P(a \mid X, h_t)$ that receives only the instruction, shared prefix, and candidate action. It accepts exactly the action that satisfies the edited requirement:

$$P(a_A \mid X_A, h_t) = P(a_B \mid X_B, h_t) = 1, \qquad P(a_B \mid X_A, h_t) = P(a_A \mid X_B, h_t) = 0. \tag{2}$$

(C2) establishes that the two trajectories succeed under different instructions, but it does not show that the divergent action at step t is what makes them differ in correctness. (C3) provides this action-level check: given the shared prefix, a_A satisfies the edited requirement only under X_A and a_B only under X_B. Our primary evaluation therefore uses groups satisfying (C1)–(C3). Groups satisfying only (C1)–(C2) are used only for full-trajectory evaluation.

Evaluation metric. In each row X_i, we convert the verifier output into a choice $\hat{a}_i \in \{a_A, a_B\}$ or a tie (§3.4); when the verifier judges complete trajectories, $\hat{a}_i$ denotes the trajectory that contains the chosen action. A group is solved only if both rows are correct:

$$\text{both_correct}(G) = \mathbf{1}\big[\hat{a}_A = a_A\big] \cdot \mathbf{1}\big[\hat{a}_B = a_B\big], \tag{3}$$

with a tie in either row counted as failure. Under one independent random choice per row, the chance is $1/4$. For the two-order pairwise protocol (§3.4), the corresponding strict chance level is $1/16$.

Table 1: Categories of crossed groups based on how the correct argument is obtained.

Class	How target is obtained	Example
Stated value	Read directly from the instruction	"...tomorrow *5PM*" ⇒ `hour=17`
Fixed mapping	Apply a fixed phrase-to-value mapping	"...I created *yesterday*" ⇒ `days=-1`
Observation lookup	Read a requested item from an observation	"the *last* stock on the list" ⇒ observed watchlist → `NVDA`
Selection by attribute	Compare observed items by an attribute	"the *cheapest* flight" ⇒ minimum observed price → `flight_number`
Observation-based computation	Compute the target from an observed value	"...*tomorrow* 5PM" ⇒ clock (Sep 16) +1 day → `day=17`

What the crossed test rules out. The second instruction rules out a fixed preference, *i.e.*, preferring the same candidate under both instructions regardless of the changed requirement. For example, a verifier that always prefers a_A can be correct under X_A but must fail under X_B. More generally, if a verifier's score decomposes as $V(T \mid X) = f(T) + g(X)$, then $g(X)$ shifts both candidates equally within a row, so the verifier selects the same candidate under both instructions. Thus, it cannot solve a crossed group where the correct candidate reverses. This only rules out a verifier that prefers the same candidate under both instructions. Other simple strategies can still succeed. For example, some groups can be solved by matching a value included in the instruction to the candidate action. §3.2 therefore groups examples by how the correct argument is obtained.

3.2 How the target argument is determined

Not all crossed groups require the same information to identify the correct action. In some groups, the target value is stated directly in the instruction; in others, it must be obtained from a mapping, read from an observation, selected from a set of observed items, or computed from observed values. We separate these cases because verifier performance has different interpretations across them. For example, success on a stated-value group may come from direct string matching, whereas success on an observation-dependent group requires using information from the trajectory prefix.

Table 1 classifies groups by how the edited argument is determined: directly from the instruction, through a fixed mapping, by reading or selecting from an observation, or by computing over an observation. For each group, we specify a derivation of the target argument using only the instruction, observations in h_t, and benchmark-provided mappings. The same derivation must reproduce the reference under both instructions; Appendix D gives the assignment rule. Classes describe how we derive the target value, not which evidence a verifier uses. We therefore separately test surface separability and whether the predicate succeeds without prefix observations.

3.3 Constructing and Validating Crossed Groups

Construction. We start from a benchmark task with a successful trajectory T_A under instruction X_A. For each editable requirement included in our construction set, we modify that requirement to obtain X_B, recompute the corresponding argument, and replace the original action a_A with a_B. We then update only downstream tool calls that depend on the edited value, producing T_B.

Validation. Before evaluating verifiers with crossed tests, we first automatically validate that each crossed group has intended ground-truth labels. These checks ensure that each group satisfies the construction requirements (C1)-(C3). We cover how verifiers are evaluated in §3.4. We first check (C1): the two instructions differ only in the edited requirement and all mentions of that requirement are updated consistently. The candidate actions must use the same function and differ only in the corresponding argument. To validate (C2), we replay both trajectories under both instructions in the environment to calculate the environment reward. We keep a group only if both trajectories execute without error and the environment assigns reward 1 exactly to the matched pairs, (X_A, T_A) and (X_B, T_B). Finally (C3), we verify the divergent action itself, rather than the final task outcome. Given the instruction, shared prefix, and a candidate action, a separate action-level predicate recomputes the

required argument and checks whether the candidate uses the correct value. We keep a group only if the predicate selects a_A under X_A and a_B under X_B.

3.4 Protocols for Evaluating Verifiers

After validating the crossed groups, we evaluate how verifiers score the two candidate actions or trajectories. Unlike the checks in §3.3, the verifier does not receive ground-truth labels or validation predicates, and each instruction is evaluated independently.

We use four protocols. Decision-point pairwise, our primary protocol, presents the shared prefix and both candidate actions and asks for a direct preference in both presentation orders. Decision-point pointwise scores each action separately and selects the higher-scored one. Full-trajectory step labeling assigns a score to each step and aggregates the labels by their minimum or mean. Full-trajectory pairwise directly compares the two complete trajectories. Decision-point pointwise and pairwise differ only in whether candidates are scored separately or compared in the same call; minimum and mean use the same step labels. A group is both correct only when the verifier selects the matched candidate under both instructions; ties or unresolved outputs count as failures. We additionally distinguish fixed preference, position-following, and other failure forms in the analysis, and report committed error rate (CER) when relevant. For step-label protocols, we also report detection at the edited step and false alarms on matched trajectories.

4 Experiments

We organize the experiments around three questions. First, what does the crossed test reveal beyond evaluating the original instruction alone? Second, can the diagnosed selection failures be reduced by changing how the observed options are presented? Third, how do scoring and aggregation choices affect whether local judgments produce a correct ranking?

4.1 Setup

Benchmarks. We construct 195 groups from three benchmarks with executable environments and task rewards: BFCL multi-turn (Patil et al., 2025), ToolSandbox (Lu et al., 2024), and τ^2-bench (Barres et al., 2026). The benchmark and relation-class breakdown is in Table 16. Every group passes the execution, rendering, and predicate checks of §3.3. A blinded human audit of 100 rows agrees with the constructed action on $91/100$ rows by majority vote (91.0%, 95% CI $[83.8, 95.2]$), with Gwet's AC1 of 0.716 (Appendix F). We exclude 27 free-text substitutions from the headline results and report two boolean-polarity pilots separately. Construction sources and exclusions are documented in Appendices B, C and E.

Verifiers. Prompted judges are deepseek-v4.1-flash, Qwen3.8-27B, Qwen3.5-35B-A3B, GPT-5.6 Luna, and GPT-5.6 Terra. The first three are evaluated under all four protocols of §3.4; the two GPT judges are added to the selection experiments. Trained verifiers are ToolRM-Gen 4B, ToolPRM-GRPO v3 and v4, Skywork-Reward-V2 8B, and Agent-RRM 8B, each evaluated in its native format. Qwen2.5-Math-PRM-7B is a mismatched-domain control. Prompts, decoding settings, and model availability are reported in Appendices J and K.

Controls. Trivial instruction-independent scorers fail the crossed test, and stated-value groups serve as a protocol sanity check. A string-matching scorer succeeds only when the differing action value is exposed in the instruction and fails on the non-literal relation classes. Full controls and baseline results are in Appendices D and O and Table 19.

4.2 Crossed tests reveal failures hidden by single-instruction evaluation

Selection-by-attribute groups provide the clearest case where a verifier appears correct under the original instruction but fails when the required attribute changes. These groups change only the attribute used to choose an item from a listing. We construct 15 groups from five scenario clusters spanning airline, retail, and reminder domains, with criterion changes such as cheapest versus earliest, fastest, or most expensive. Each group is evaluated three times.

Table 2: Failures exposed by the second instruction on 15 selection groups (first pairwise call). The middle column shows groups correct under the original instruction and the number remaining correct when both instructions are required. The final column classifies only the newly exposed failures.

Judge	Correct groups: original → both	Newly exposed failures
deepseek-v4.1-flash	$10 \rightarrow 7$	1 fixed; 2 unresolved
Qwen3.8-27B	$7 \rightarrow 6$	1 fixed
Qwen3.5-35B-A3B	$8 \rightarrow 5$	3 fixed
GPT-5.6 Luna	$6 \rightarrow 5$	1 fixed
GPT-5.6 Terra	$11 \rightarrow 7$	1 fixed; 2 position; 1 tie
Total	$\mathbf{42 \rightarrow 30}$	**7 fixed; 2 position; 1 tie; 2 unresolved**

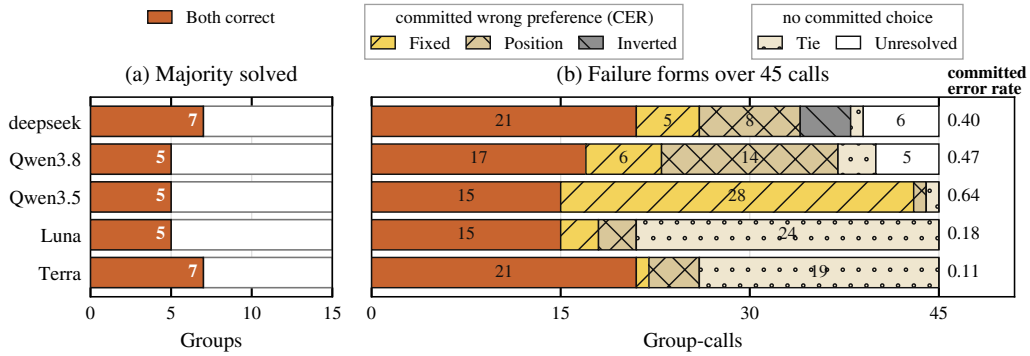

Figure 3: Selection-by-attribute results on 15 groups evaluated with three calls. **(a)** Groups solved by a majority of calls. **(b)** Outcomes of the 45 group-calls per judge, decomposed into both-correct, fixed-preference, position-following, inverted, tie, and unresolved outcomes. CER summarizes committed wrong preferences. Similar both-correct counts can reflect different failure modes.

What the second instruction adds. We compare the original-instruction row with the strict crossed score using the first pairwise call (Table 2). Across the five judges, 42 judge–group pairs are correct under the original instruction, of which 30 are also correct under the edited instruction. The reduction alone reflects the stricter two-instruction requirement; the more informative signal is how the 12 additional failures occur. They include seven fixed preferences, two position-following choices, one tie, and two unresolved outputs. The crossed test also identifies fixed preferences in groups whose original row was already wrong; a single-row score counts these only as ordinary errors.

Different judges fail in different ways. The five judges obtain similar crossed scores, but their failure modes differ sharply (Figure 3). Qwen3.5-35B-A3B is dominated by fixed preferences, Qwen3.8-27B by position-following, while the GPT-5.6 judges more often return ties. Consequently, CER ranges from 0.11 to 0.64 despite similar both-correct counts. These distinctions matter operationally: a fixed or position-following choice supplies incorrect preference supervision, whereas a tie abstains. A fixed preference indicates failure to reverse for the tested requirement change, not that the judge generally ignores instructions. Appendix Q gives the construction, tie subtypes, and per-call analysis.

The failures persist beyond the original prefixes and listings. Adding policy-required authentication and confirmation to the τ^2-bench prefixes leaves the pairwise majority outcome unchanged for deepseek-v4.1-flash and Qwen3.5-35B-A3B, while Qwen3.8-27B improves from 1 to 3 of 11 selection groups (Appendix I). Thus, the main pattern is not explained by the omitted policy steps.

The same pattern appears on 11 held-out groups from six new listings in the same domains. At the original input, no judge solves more than $2/11$ groups by majority. Withholding the listing reduces every judge to $0/15$ on the diagnosed groups and $0/11$ on the held-out groups (Appendix Q), confirming that the observed listing is necessary for these evaluations. Across all 113 evaluated

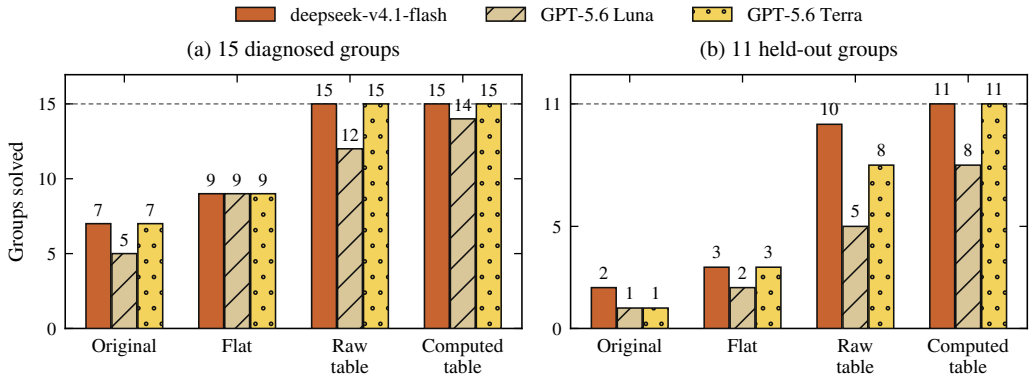

Figure 4: Selection performance under four renderings. Flat reformats the original listing into rows; raw table groups observed attributes by item without computing new values; computed table additionally provides derived attributes. Bars show groups solved by a majority of three calls on **(a)** 15 diagnosed and **(b)** 11 held-out groups; dashed lines mark the number of groups in each set.

groups for the original three judges, no originally correct group outside selection fails the second row in this run. Together, these controls show that the selection failures are not explained by the omitted policy steps or by idiosyncrasies of the original listings.

4.3 Organizing observed options improves selection

The failures above could arise because the relevant information is absent, or because it is difficult to compare in the original listing format. We therefore keep the instruction and candidate actions fixed and change only how the observed options are presented. We compare three increasingly structured renderings against the original listing for deepseek-v4.1-flash and the two GPT-5.6 judges; the original input and computed table are available for all five judges (Figure 4 and Table 12). Flat only reformats the observation by placing each leaf object and its original fields in a row. Raw table groups attributes by item and retains the observed values relevant to the candidate configuration, such as prices for the booked cabin, without computing new values. Computed table additionally provides derived attributes used by the selection criterion, such as total price and duration. On both diagnosed and held-out groups, flat formatting provides only modest gains, while grouping attributes by item recovers most failures; computed tables provide a further improvement on the held-out set. Thus, grouping the compared items and retaining their relevant observed attributes accounts for most of the recovery without supplying computed values.

The recovery depends on how the options are organized, not merely on converting the listing into a table. Flat reaches ceiling on the single-object retail groups but remains weak on multi-leg airline listings, while grouping legs by itinerary accounts for most of the gain (Appendix Q). Because the crossed score requires correct choices under both requirements, the improvement cannot arise from strengthening a fixed preference for one candidate. This is consistent with prior findings that LLM reasoning over structured data is sensitive to serialization and structural presentation (Sui et al., 2024). The held-out recovery also persists under an equal output budget: deepseek-v4.1-flash solves $2/11$ groups with the original listing and $11/11$ with the computed table (Appendix Q).

4.4 Scoring and aggregation can obscure correct local judgments

Even when a verifier distinguishes the edited action locally, the scoring protocol can fail to turn that judgment into the correct ranking. Figure 5 examines two such cases: scoring candidates separately rather than comparing them directly, and aggregating step labels into a trajectory score.

Separate scoring versus direct comparison. The two Qwen judges improve under pairwise comparison on both stated values and hour conversion, while deepseek-v4.1-flash changes little. Across the pooled 92-group comparison, pairwise scoring produces many more gains than losses for

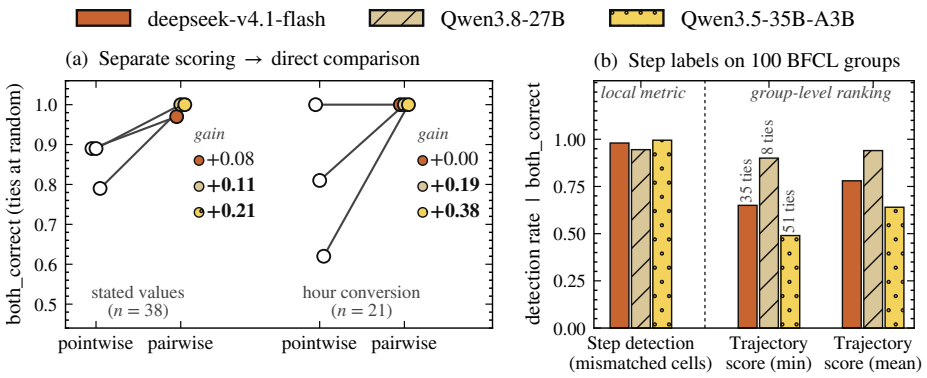

Figure 5: **(a)** Decision-point both-correct accuracy under pointwise scoring (open) and pairwise comparison (filled) on the same groups; numbers show the pairwise gain for each judge. Ties are broken at random and unresolved outputs count as failures. **(b)** On 100 BFCL groups, step detection is the fraction of mismatched cells labeled -1 at the edited step, while min and mean show group-level both-correct scores from the same step labels.

both Qwen judges (Figure 5). Separate scoring can assign both candidates low scores even when direct comparison selects correctly. Both protocols receive the same instruction, prefix, and candidate actions, but pairwise evaluation exposes the alternative in the same call. The gain therefore measures a protocol effect and does not establish that the judge recovered the full target derivation. Results for the remaining classes and trained verifiers are in Figure 12 and Table 19.

Local detection versus trajectory ranking. All three judges detect the edited error in at least 94.5% of mismatched cells with parseable labels, yet minimum aggregation yields substantially lower strict both-correct scores. The gap is largest for Qwen3.5-35B-A3B: detection is 0.995, but 51 tied groups reduce the score to 0.49. False alarms on correct trajectories can therefore erase the distinction created at the edited step. Applying the mean to the same labels improves all three judges, including deepseek-v4.1-flash from 0.65 to 0.78. Thus, high local detection does not ensure a correct trajectory ranking. The same aggregation effect appears for a trained prefix scorer. On the same 100 groups, Skywork-Reward-V2 8B rises from 0.81 under minimum aggregation to 0.98 under mean aggregation of its prefix scores. Agent-RRM 8B, which outputs a single trajectory score and therefore has no aggregation choice, obtains 0.48.

5 Conclusion

We introduced crossed instruction–trajectory matching to test whether agent process verifiers change their candidate preference when a task requirement changes. Execution checks and an independent action-level predicate establish the required reversal. On selection-by-attribute groups, judges with similar accuracy fail through different mixtures of fixed preference, position-following, and abstention. Organizing observed attributes by item recovers most of these failures without supplying computed values. Complementary comparisons show that separate scoring and minimum aggregation can prevent local judgments from supporting a correct ranking. The resulting evaluation separates whether a candidate was chosen correctly from whether the verifier responds correctly to a changed requirement.

Ethics Statement

This work evaluates process verifiers on tasks derived from three public tool-use benchmarks, BFCL multi-turn (Patil et al., 2025), ToolSandbox (Lu et al., 2024), and τ^2-bench (Barres et al., 2026), which we use under their released licenses. The instructions, tool environments, and user records in these benchmarks are synthetic, and our edits change a single requirement (such as a date, a destination folder, or a selection criterion) without introducing personal data. The blinded audit

(Appendix F) was conducted with 54 raters recruited through Prolific, who were compensated at a rate of $12 per hour. Raters select which of two candidate actions an instruction requires; the task involves no sensitive content, and we collect no personal information beyond anonymous platform identifiers. Our findings identify conditions under which verifiers assign incorrect preferences, and we report them to support more reliable verifier evaluation and training. We do not foresee a direct path by which these results enable harm. We declare no conflicts of interest or sponsorship that bear on this work.

Reproducibility Statement

§3 defines the crossed test, the scoring rule, and the validation checks; Appendices D to F give the relation classes, the edit forms, and the construction, predicate, and audit procedures. Appendix K lists the verifier prompts, and Appendix J lists the evaluated judges and trained verifiers with their released versions. Run settings, including the number of repeated calls, output budgets, and tie handling, are given with each experiment in Appendices I and Q. Because API-served judges are not deterministic, the selection experiments report majority outcomes over three byte-identical calls, and Appendix O reports the effect of repeated calls on the small classes. The supplementary material contains the constructed groups with their four execution-graded cells, the predicate and evaluation code, and the scripts that produce every figure and table from stored result files.

AI Use Statement

We used generative AI tools for the following tasks. *Implementing methods:* AI coding assistants helped implement parts of the experimental code, including the group construction and validation pipeline, the verifier evaluation harness, and the analysis scripts. *Creating figures and tables:* AI tools helped organize experimental results into figures and tables, including writing the plotting scripts. *Editing the paper:* AI tools helped revise drafts of the paper for readability, consistency of terminology, and alignment between figures, captions, and text. AI tools were also used for a secondary qualitative coding of verifier rationales reported in Appendix O; the coding scheme and aggregate counts were reviewed by the authors. The LLM judges and trained reward models evaluated in Sections 4.2 and 4.4 are objects of study rather than tools used to conduct the research. The authors reviewed all AI-assisted work: AI-written code was inspected and tested by the authors, the numbers reported in figures and tables were checked against the underlying result files, and all edited text was reviewed. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

References

Mayank Agarwal, Ibrahim Abdelaziz, Kinjal Basu, Merve Unuvar, Luis A. Lastras, Yara Rizk, and Pavan Kapanipathi. ToolRM: Outcome reward models for tool-calling large language models. *arXiv preprint arXiv:2509.11963*, 2025.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik R Narasimhan. $\tauˆ2$-bench: Evaluating conversational agents in a dual-control environment. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=OC2z7iSQKa.

Hyungjoo Chae, Sunghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, Dongwook Choi, Minseok Kang, Gwanhoon Im, ByeongUng Cho, Hyojun Kim, Jun Hee Han, Taeyoon Kwon, Minju Kim, Beong woo Kwak, Dongjin Kang, and Jinyoung Yeo. Web-shepherd: Advancing PRMs for reinforcing web agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=G2kMroO9UV.

Tsong Yueh Chen, Shing Chi Cheung, and Shiu Ming Yiu. Metamorphic testing: A new approach for generating next test cases. Technical Report HKUST-CS98-01, Hong Kong University of Science and Technology, 1998.

Steven Cho, Stefano Ruberto, and Valerio Terragni. Metamorphic testing of large language models for natural language processing. In *2025 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 174–186. IEEE, 2025.

Sanjiban Choudhury. Process reward models for LLM agents: Practical framework and directions. *arXiv preprint arXiv:2502.10325*, 2025.

Shengda Fan, Xuyan Ye, Yupeng Huo, Zhi-Yuan Chen, Yiju Guo, Shenzhi Yang, Wenkai Yang, Shuqi Ye, Jingwen Chen, Haotian Chen, et al. Agentprocessbench: Diagnosing step-level process quality in tool-using agents. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 8823–8834, 2026.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. *International Conference on Machine Learning*, 2023.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, et al. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1307–1323, 2020.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1755–1797, 2025.

Dawei Li, Yuguang Yao, Zhen Tan, Huan Liu, and Ruocheng Guo. Toolprmbench: Evaluating and advancing process reward models for tool-using agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12378–12391, 2026.

Renhao Li, Jianhong Tu, Yang Su, Yantao Liu, Fei Huang, Hamid Alinejad-Rokny, Derek F. Wong, Junyang Lin, and Min Yang. ToolRM: Towards agentic tool-use reward modeling. *arXiv preprint arXiv:2510.26167*, 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. *arXiv preprint arXiv:2305.20050*, 2023.

Jianghao Lin, Yuanyuan Shi, Xin Peng, Renjie Ding, Hairui Wang, Yuxuan Peng, Bizhe Bai, Weixi Song, Fengshuo Bai, Huacan Chai, Weinan Zhang, Fei Huang, and Ying Wen. ToolPRM: Fine-grained inference scaling of structured outputs for function calling. *arXiv preprint arXiv:2510.14703*, 2025.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Felix Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. *arXiv preprint arXiv:2408.04682*, 2024.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stanczak, Peter Shaw, Christopher Pal, and Siva Reddy. Agentrewardbench: Evaluating automatic evaluations of web agent trajectories. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=fQcUZMPIvu.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 4902–4912, 2020.

Lingfeng Shen, Sihao Chen, Linfeng Song, Lifeng Jin, Baolin Peng, Haitao Mi, Daniel Khashabi, and Dong Yu. The trickle-down impact of reward inconsistency on rlhf. In *International Conference on Learning Representations*, 2024.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic study of position bias in llm-as-a-judge. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 292–314, 2025.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. Prmbench: A fine-grained and challenging benchmark for process-level reward models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25299–25346, 2025.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 645–654, 2024.

Rishabh Tiwari, Aditya Tomar, Udbhav Bamba, Monishwaran Maheswaran, Heng Yang, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. Reward under attack: Analyzing the robustness and hackability of process reward models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=phHnze3Nfm.

Tuhina Tripathi, Manya Wadhwa, Greg Durrett, and Scott Niekum. Pairwise or pointwise? evaluating feedback protocols for bias in LLM-based evaluation. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=uyX5Vnow3U.

Atticus Wang, Iván Arcuschin, and Arthur Conmy. Automatically finding reward model biases. In *Algorithmic Fairness Across Alignment Procedures and Agentic Systems*, 2026a. URL https://openreview.net/forum?id=G7nQsCVy4x.

Bowen Wang, Dunjie Lu, Junli Wang, Tianyi Bai, Shixuan Liu, Zhipeng Zhang, Haiquan Wang, Hao Hu, Tianbao Xie, Shuai Bai, et al. Cua-gym: Scaling verifiable training environments and tasks for computer-use agents. *arXiv preprint arXiv:2605.25624*, 2026b.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, et al. Large language models are not fair evaluators. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 9440–9450, 2024a.

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*, 2024b.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, et al. Agentprm: Process reward models for llm agents via step-wise promise and progress. In *Proceedings of the ACM Web Conference 2026*, pp. 4184–4195, 2026.

Ke Yang, Jiateng Liu, John Wu, Chaoqi Yang, Yi R Fung, Sha Li, Zixuan Huang, Xu Cao, Xingyao Wang, Yiquan Wang, et al. If llm is the wizard, then code is the wand: A survey on how code empowers large language models to serve as intelligent agents. *arXiv preprint arXiv:2401.00812*, 2024.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025.

Ziyun Zhang, Zezhou Wang, Xiaoyi Zhang, Zongyu Guo, Jiahao Li, Bin Li, and Yan Lu. Infiniteweb: Scalable web environment synthesis for gui agent training. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28465–28492, 2026.

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. *arXiv preprint arXiv:2412.06559*, 2024.

APPENDIX CONTENTS

A Discussion and Limitations **15**

B Benchmark integration and benchmark-level findings **15**

C Boolean polarity pilot **15**

D Class assignment and pair labels **16**

E Edit forms **16**

F Construction and audit details **17**

G Measurement traps **18**

G.1 Rendering errors and diagnostic comparisons 18

H Relation-class details **19**

I τ^2-bench prefixes and policy compliance **21**

J Baselines and availability **22**

K Prompts **22**

L Single-axis rewrites (secondary protocol) **26**

M Performance across relation classes **26**

N Whole-trajectory and decision-point detection **27**

O Input ablation details **27**

P Single-row accuracy against the crossed score **30**

Q Selection-by-criterion groups **31**

R Selection on held-out actions **34**

R.1 Targeted clock set 36

S Additional experimental tables and figures **37**

S.1 Inventory and observation removal 37

S.2 Clock representation 37

S.3 Scoring and aggregation 37

T Human audit interface **37**

A Discussion and Limitations

Coverage and validity. The construction requires executable tasks with checkers and consistently editable references. The inventory is dominated by BFCL stated-value groups, many of which a surface matcher can solve. The selection results cover 15 groups from five scenario clusters, and the held-out replication of the repair uses new listings of the same two domains and criterion pairs, so it shows that the repair holds on new cases of this decision type and not that it extends to other kinds of selection; several other non-literal families contain only a few groups. These results characterize the tested decisions rather than the prevalence of each failure in policy rollouts. All 195 groups pass the automated checks. In a blinded audit of 100 rows, human majority judgments agree with the constructed action on 91.0% of rows (Gwet's AC1 $= 0.716$; Appendix F). The remaining non-matching rows are primarily abstentions or lack a strict majority rather than showing a systematic majority preference for the alternative candidate.

Evaluation scope. In τ^2-bench, gold prefixes omit policy-required steps. Adding the required authentication call and confirmation turn leaves the majority outcome unchanged for two judges and raises Qwen3.8-27B from 1 to 3 of 11 selection groups; the selection failures persist (Appendix I). The full-trajectory step-label read-out remains excluded for τ^2-bench. Protocol comparisons also change candidate visibility or output format and must be interpreted under the stated scoring rule. Additional input ablations and exploratory policy-generated selection experiments are reported in Appendices O and R; they do not establish general downstream gains.

B Benchmark integration and benchmark-level findings

Pins and fixes. BFCL is pinned to gorilla `6ea57973`; the checker caches backend instances globally by (model name, entry id), so two replays in one process share state. We wrap each replay with a unique name and delete the instance afterwards. τ^2-bench is pinned to `2174a603` with a one-line local patch: at this pin the scripted user does not accept the arguments the runner passes in solo mode.

Judged grading in τ^2-bench retail. Forty retail tasks contain natural-language assertions graded by an LLM. Thirty-two assert that the agent stated a specific value (a refund total, a gift-card balance); eight assert that an action happened, which the database half already decides. The judge (gpt-4.1) passes $30/32$ stated-value assertions when the value is stated and fails all 32 when it is omitted. The two failures are the judge disputing the benchmark's own assertion ($481.50 in the assertion against $46.66 in the order data). We use $74 + 30 = 104$ retail tasks and exclude the eight action-only tasks, because grading them from the database alone would change the benchmark's reward definition. The probe was revised once (assertions rewritten into plain agent statements) after an initial pass rate of $4/40$.

Empty-reply pass. τ^2-bench scores judged tasks as `all(check.met for check in checks)`; an empty agent reply produces zero checks and therefore reward 1. We add a guard that raises on an empty check list and log per-cell verdicts.

C Boolean polarity pilot

Polarity groups flip the target state of a setter (`on`↔`off`) together with the initial state, because an instruction that is already satisfied by the initial state filters nothing (construction rule on satisfied initial states). ToolSandbox yields two natural groups (cellular off, wifi off) and four semi-natural ones in the low-battery family whose X_B world state is inconsistent (location on while low-power mode is on). τ^2-bench telecom exposes only toggle tools to the agent, so one call satisfies both polarities and $T_A = T_B$ (identical-trajectory rule). Because a toggle call has no polarity, polarity can only be read from the initial state, which makes the initial-state cue structurally unavoidable in telecom. With two natural groups the class is reported as a pilot; both large judges obtain $2/2$ under P5.

D CLASS ASSIGNMENT AND PAIR LABELS

Derivation record. For every group we record $v = d(s, o_{<t}, \mathcal{T})$, where v is the edited argument, s the span that states the edited requirement, $o_{<t}$ the observations in h_t, and $\mathcal{T}$ fixed tables published with the benchmark (aliases, enumerations). The reference argument is not an input, and the same d must reproduce the reference under both X_A and X_B. Source spans, tables, and operations are recorded per group.

Assigning a class. The class is the outermost constructor of d. When several derivations reproduce the reference under both instructions, a derivation that uses no observation takes precedence, and the group takes the first class in the order stated value, fixed mapping that reproduces the reference; otherwise it takes the first class in the order observation lookup, selection by attribute, observation-based computation. A change of notation denotes the same object in another format ("5 PM" and `hour=17`; a benchmark alias and its canonical name) and stays in the stated-value class. A fixed-mapping table assigns a conventional value to a phrase ("yesterday" and an offset of -1 day). An observation-based computation produces a value that appears in none of its inputs, such as the calendar day that "tomorrow" or "next Friday" denotes given the observed clock. The inventory contains no group whose value is computed from stated values alone, so no separate class is defined for that case. The order is a tie-break rule and carries no claim about difficulty.

String-matching scorer. The scorer extracts the tokens in which a_A and a_B differ and selects the action whose differing tokens occur in the instruction; digits match only as whole numbers, so that `2` does not match inside "May 27". An order-aware variant compares the order in which two values are first mentioned in the instruction with their order in the call. A pair is surface-separable if either variant selects the correct action in both rows. On the inventory the scorer solves 0.90 of the stated-value groups, none of the fixed-mapping, list-index, selection-by-attribute, and observation-based computation groups, and all six groups of the τ^2-bench payment-method family, in which the identifier read from the user record contains the payment type named in the instruction (`paypal_...`, `gift_card_...`). That family keeps its class, observation lookup, and is reported as surface-separable. In 11 of the 21 stated-value hour groups one row is matched by digits ("8AM" and `hour=8`).

Observation-blind predicate. A pair that is not surface-separable is observation-free if a variant of P that receives X and the two candidates, without the observations of h_t, still accepts exactly the matched actions. For the clock families this variant checks whether the weekday of each candidate date agrees with the weekday named in the instruction. Under "next Friday" $\rightarrow$ "next Monday" the candidate dates are September 18 and 21, 2026, and the check separates them; under "tomorrow" $\rightarrow$ "the day after tomorrow" the candidates are September 17 and 18 and the check has nothing to compare, so this routine leaves the pair unresolved without the observation. We use observation-dependence labels operationally; predicate failure alone is not proof that the observation is necessary. Run without the observations, the predicate reproduces the matched actions in all 147 groups of the stated-value and fixed-mapping classes and the six weekday groups, in each of which the two candidate dates fall on different weekdays, and in none of the 18 list-index, 8 date (3 epoch, 5 ISO), 15 selection-by-attribute, 1 baggage, and 6 payment-method groups; the payment-method family is surface-separable and keeps the label S. The weekday groups keep the class observation-based computation, since the reference derivation reads the clock, and carry the pair label F.

E EDIT FORMS

Every group changes one requirement. In 165 groups the requirement is stated in one span. In 19 it is mentioned more than once ("the third file" and later "that same third file"), and in 11 it has several surface forms that must agree (file name and extension, company name and ticker, airport code and city, a count and its noun). A leftover check rejects a group if any mention of the old value survives in X_B. Nine further BFCL groups exchange two arguments of one call ("from X to Y"); they violate the single-argument condition of (C1), are solved by the order-aware string scorer, and are reported as a sensitivity set.

F CONSTRUCTION AND AUDIT DETAILS

Construction. The replacement value is drawn from the same benchmark domain as the original. Each of the four cells is graded twice, and three times when the benchmark grades a natural-language assertion with an LLM; groups with inconsistent labels are rejected. The checker implementation is unchanged, while the reference it grades against is edited to reflect the new requirement. The predicate routines are: equality up to notation for stated values, the phrase table for fixed mappings, the position in the observed list for observation lookups, a comparison over all observed prices or departure times for selection by attribute, and an independent calendar routine for observation-based computations. The predicate holds on all 195 groups, and exchanging X_A and X_B reverses its output in every group.

Rendering and audit. Machine items (i)–(iii) pass on all headline groups: (i) mirrored cells under the benchmark checker; (ii) argument names rendered with their values (the earlier criterion, "both endpoints appear in the input", was vacuous because both endpoints appear in both trajectories of a role swap); (iii) the rendered line diff between T_A and T_B classified as endpoint / numeric consequence / other, with zero "other" lines. Step counts and error counts are equal in every group and rendered lengths differ by at most two characters.

For the blinded semantic audit, 54 raters were recruited through Prolific and compensated at a rate of $12 per hour. Raters saw the instruction, shared prefix with its observations, and the two candidate actions in randomized order, without verifier scores, checker labels, or the construction reference. They selected which action the instruction called for, with additional options for both actions, neither action, or insufficient information; they also judged whether the history was a reasonable way to begin the task and reported confidence. Each row's label is the answer of a strict majority of its valid raters; rows with fewer than two valid raters are excluded from majority scoring.

Across 100 audited rows, the majority agrees with the constructed action on 91 rows (91.0%, 95% CI $[83.8, 95.2]$). Agreement is $42/48$ (87.5%, 95% CI $[75.3, 94.1]$) for observation-lookup, observation-based computation, and fixed-mapping groups, $28/30$ (93.3%, 95% CI $[78.7, 98.2]$) for diagnosed selection groups, and $21/22$ (95.5%, 95% CI $[78.2, 99.2]$) for held-out selection groups. Inter-rater agreement measured by Gwet's AC1 is 0.716. Across the 300 individual ratings, 252 select the constructed action, 14 the other candidate, 1 both candidates, 20 neither candidate, and 13 cannot determine the answer. At the row level, the nine rows without a constructed-action majority comprise one "neither" majority, three "cannot determine" majorities, and five rows with no strict majority; no row has a majority for the alternative candidate or for both candidates.

Table 3: Blinded human audit. A row matches when a strict majority of valid raters selects the constructed action; rows with fewer than two valid ratings are excluded.

Stratum	Rows	Match	Accuracy	95% CI
All	100	91	91.0%	[83.8, 95.2]
Observation lookup / computation / fixed mapping	48	42	87.5%	[75.3, 94.1]
Selection, diagnosed	30	28	93.3%	[78.7, 98.2]
Selection, held-out	22	21	95.5%	[78.2, 99.2]

Sensitivity to the audit. The nine rows without a constructed-action majority belong to seven groups: two diagnosed and one held-out selection group, three list-index groups, and one τ^2-bench ISO-date group. We recompute the headline results without these seven groups from the stored verifier calls, without new model calls (Table 4). On the diagnosed selection groups, the dominant failure form of each judge is unchanged: Qwen3.5-35B-A3B's fixed preferences fall from 28 of 45 to 23 of 39 group-calls, Qwen3.8-27B's position-following from 14 of 45 to 13 of 39, and ties remain the most frequent failure of GPT-5.6 Luna (24 of 45 to 19 of 39) and Terra (19 of 45 to 18 of 39). The failures exposed by the second instruction (Table 2) become 40 groups correct under the original instruction and 29 under both, with 11 newly exposed failures: six fixed preferences, two position-following choices, one tie, and two unresolved outputs. In the renderer controls (Figure 4), flat solves 8 of the 13 remaining diagnosed groups for deepseek-v4.1-flash and both GPT-5.6 judges, and raw table solves 13, 10, and 13; on the 10 remaining held-out groups, flat solves 3, 2, and 3

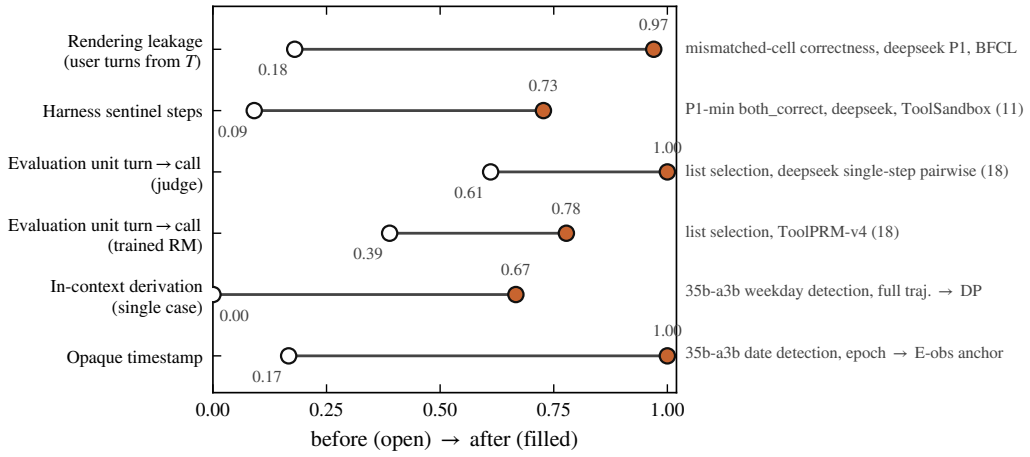

Figure 6: Measurement traps: the same verifier on the same groups before (hollow) and after (filled) the pipeline fix or the change of regime. Rows 1–2 are rendering errors; rows 3–4 are the evaluation unit; row 5 is the one case in which detection itself differs between whole-trajectory labelling and the decision point (Qwen3.5-35B-A3B, weekday); row 6 is the observation as rendered against the observation with the clock written out as an anchor.

and raw table 9, 5, and 7. Under the equal-budget control, the original listing and the computed table solve $2/10$ and $10/10$ groups for deepseek-v4.1-flash and $1/10$ and $10/10$ for Qwen3.8-27B. The four excluded list-index and ISO-date groups are solved under pairwise comparison by all three judges in Figure 12; without them, pairwise comparison remains at or above pointwise scoring in every class, and each judge's count of groups solved only under pairwise comparison decreases by at most one. Across the cells reported in Figures 3 and 4 and Tables 2 and 12, the share of groups solved by majority changes by -3 to $+7$ percentage points and CER by -6 to $+5$ points. Excluding these groups therefore does not change the qualitative conclusions.

Table 4: Selection results on all groups and after excluding the groups that contain an audited row without a constructed-action majority (all → remaining). Groups solved by a majority of three calls; CER in the diagnosed original-input condition. Held-out original values for deepseek-v4.1-flash and Qwen3.8-27B are from the equal-budget rerun, as in Table 12.

	Diagnosed (15 → 13)			Held-out (11 → 10)	
Judge	Original	CER	Table	Original	Table
deepseek-v4.1-flash	7 → 6	.40 → .38	15 → 13	2 → 2	11 → 10
Qwen3.8-27B	5 → 5	.47 → .51	15 → 13	1 → 1	11 → 10
Qwen3.5-35B-A3B	5 → 5	.64 → .62	15 → 13	1 → 1	11 → 10
GPT-5.6 Luna	5 → 5	.18 → .15	14 → 12	1 → 1	8 → 8
GPT-5.6 Terra	7 → 6	.11 → .05	15 → 13	1 → 1	11 → 10

G MEASUREMENT TRAPS

Table 5 lists every construction and measurement error found, the check that detected it, and the metric change on the same groups.

G.1 RENDERING ERRORS AND DIAGNOSTIC COMPARISONS

Figure 6 and Table 5 list the errors. The three withdrawn results were a nine-pair role-swap asymmetry (rendering leakage), a precision ordering of judges on ToolSandbox (harness sentinels rendered as steps), and a $0/12$ detection collapse of Qwen3.5-35B-A3B on the weekday pairs under whole-trajectory labelling, which the decision-point evaluation reduced to a single-family difference

Table 5: Construction and measurement traps, the check that detected each, and the metric change on the same groups. Effects are counts on the BFCL and ToolSandbox censuses.

Trap	Detected by	Metric	Before → after
User turns rendered from T	rendering check	BFCL mismatched-cell correctness, deepseek-v4.1-flash P1; $\mathrm{both_correct}$	$0.18 \rightarrow 0.97$; $0.06 \rightarrow 0.65$
Harness sentinels rendered as steps	step-type check	ToolSandbox step labels (min) $\mathrm{both_correct}$, deepseek-v4.1-flash/ Qwen3.8-27B (11)	$1/11 \rightarrow 8/11$; $6/11 \rightarrow 8/11$
Decision point at turn instead of call	call-level split	list selection (18), deepseek-v4.1-flash single-step; ToolRM-Gen/ ToolPRM-GRPO v4 / Skywork	$0.61 \rightarrow 1.00$; $0.67 \rightarrow 1.00$ / $0.39 \rightarrow 0.78$ / $0.44 \rightarrow 0.89$
A single pointwise number in either regime	DP loss decomposition; offline det/miss/na	whole-trajectory: matched-cell false alarms and minimum aggregation; decision point: both actions scored at the floor. Detection itself differs in one case (Qwen3.5-35B-A3B weekday)	$0/12 \rightarrow 4/6$ (that case)
Observed clock not readable	E-obs against the annotation control (same decisions); τ^2-bench ISO dates (different tasks)	Qwen3.5-35B-A3B date detection: clock written out / later argument annotated	$1/6 \rightarrow 6/6$ / $1/6 \rightarrow 1/6$; $4/5$ (ISO)
Order-dependent answers	two-order protocol	ToolPRM-GRPO v3	37 order ties
Budget exhaustion read as abstention	output-length audit	ToolPRM-GRPO v4 no-answers are 20–38k-character loops; E-obs ends them	median output $12{,}656 \rightarrow 1{,}341$ chars
Flipped call fails at execution	execution gate	24 `mv`/`cp` role swaps	rejected
Checker blind spots (τ^2-bench)	manual inspection	`all([])` passes an empty reply; gold not policy-compliant	benchmark findings
Construction bugs (value collision, format drift, time zone, days vs hours, dependent dates, unobservable inputs)	construction gates	26 ToolSandbox groups byte-identical after fixes; 16 τ^2-bench groups	–

(Appendix N). The rendering error is the most instructive. BFCL trajectories carry the user message of each turn; an early renderer took those messages from the trajectory and only the instruction header from X, so in mismatched cells the transcript read "user asks for *archive*, agent does *archive*", a step labeller consistently labelled the step $+1$, detection was 0.18, and $\mathrm{both_correct} = 0.06$ matched the independence prediction $p^2_{\text{match}}\, p^2_{\text{mismatch}}$ to two decimals. The result was caught by the rendering check, not by any statistic. The evaluation-unit rows are the empirical case for the evidence requirement of (C3): when the listing call and the selection share one turn, a turn-level decision point does not show the observation the selection depends on, and moving the decision point to the call raises deepseek-v4.1-flash's single-step accuracy on list selection from 0.61 to 1.00, ToolRM-Gen from 0.67 to 1.00, ToolPRM-GRPO v4 from 0.39 to 0.78, and Skywork (final prefix score) from 0.44 to 0.89 (Appendix H). The turn-level numbers would have been reported as a relation-class failure of three trained verifiers, whereas they measure what the evaluator withheld.

H Relation-class details

Per-family results. The computed classes come from one family (weekday), two (ToolSandbox epoch dates), and three (τ^2-bench ISO dates); hour conversion from four and list selection from six. Because families are few, we report per-family $\mathrm{both_correct}$, the paired difference $\Delta_f = \mathrm{acc}^c_f - \mathrm{acc}^{\mathrm{orig}}_f$ for each condition c (Mask, E-obs, annotation control), where acc is the accuracy metric reported with the result; group-level intervals are not reported for these classes because the groups of a family are not independent.

Full-trajectory P1 detection by class. Detected / missed / no-answer on mismatched cells, first changed step, harness steps dropped. Hour (21 groups, 42 cells): deepseek-v4.1-flash 34/0/8, Qwen3.8-27B 16/2/24, Qwen3.5-35B-A3B 36/0/6. Holiday (8 cells): 8/0/0 for all three. Map (4 cells): 4/0/0, 1/0/3, 4/0/0. Weekday (12 cells): 12/0/0, 6/1/5, 0/12/0. Date, ToolSandbox epoch (6 cells): 4/0/2, 2/2/2, 1/3/2. List selection (36 cells): 34/2/0, 36/0/0, 35/1/0. Qwen3.8-27B's no-answers are reasoning loops of 18–22k characters on time arithmetic.

Decision-point pointwise losses. Per class, both / tie / wrong-above-correct / no-answer. Qwen3.5-35B-A3B: stated $32/2/3/3$, hour $13/3/4/1$, weekday $4/2/0/0$, date $1/2/0/0$, list $17/0/0/1$. Qwen3.8-27B: stated $36/1/3/0$, hour $17/0/2/2$, weekday $4/0/1/1$. deepseek-v4.1-flash: stated $36/2/2/0$, all other classes without loss. Answered-only $\mathrm{both_correct}$ pointwise $\rightarrow$ pairwise: Qwen3.5-35B-A3B stated $0.86 \rightarrow 1.00$, hour $0.65 \rightarrow 1.00$; Qwen3.8-27B $0.90 \rightarrow 1.00$, $0.89 \rightarrow 1.00$. McNemar on group outcomes (b: pointwise fails, pairwise passes; c: reverse): Qwen3.5-35B-A3B hour $8/0$ ($p = 0.0078$, $\Delta \in [+0.11, +0.58]$), stated $8/0$ ($p = 0.0078$, $[+0.07, +0.33]$), pooled 92 groups $19/1$ ($p = 4 \times 10^{-5}$, $[+0.10, +0.28]$); Qwen3.8-27B pooled $11/0$ ($p = 0.0010$, $[+0.03, +0.21]$), stated $4/0$ ($p = 0.125$); deepseek-v4.1-flash pooled $4/1$ ($p = 0.375$). Score pattern of reversed rows (matching vs mismatched): deepseek-v4.1-flash 4 rows, all with both scores low; Qwen3.8-27B 7 rows, 5 both-low and 2 both-high; Qwen3.5-35B-A3B 22 rows, 20 both-low, 1 both-high, 1 mixed, 10 exact ties. Reasons given for scoring the matching call down: Qwen3.5-35B-A3B 14 own date miscomputations, 2 penalties on a conversion call, 1 unverified input, 5 other; Qwen3.8-27B 2 / 0 / 4 / 1; deepseek-v4.1-flash 0 / 0 / 2 / 2.

Paired conditions. Mask replaces the clock observation with `<tool output withheld>`. Mask and E-obs modify h_t and apply to all protocols; the annotation control modifies a step after t and applies to full-trajectory protocols only. In the ToolSandbox clock families the judged call is `datetime_info_to_timestamp` with readable fields (`day=17`), and the epoch value that the annotation control annotates is the argument of the later `add_reminder` call, so this control tests the effect of an annotation as such and not the notation of the judged argument. A verifier that uses the observation should lose observation-dependent pairs under Mask and keep observation-free ones. E-obs appends to the clock observation the string `(= 2026-09-16 12:00, Wednesday)`, i.e. the observed time and its weekday and nothing about the target. The annotation control renders the later call's epoch argument as `2026-09-21 17:00` with no weekday. A withdrawn earlier variant rendered it as `(= 2026-09-21 17:00, Monday)`; with "next Monday" in the instruction the check reduces to a string match on the weekday, and the $12/12$ recovery it produced for Qwen3.5-35B-A3B is not evidence about representation. Results under the annotation control (detection · matched false alarms · crossed): Qwen3.5-35B-A3B weekday $0/12 \cdot 0/12 \cdot 0/6$, date $1/6 \cdot 1/6 \cdot 0/3$; deepseek-v4.1-flash weekday $9/9 \cdot 0/12 \cdot 3/3$ (3 no-answers), date $5/5 \cdot 0/6 \cdot 2/2$; Qwen3.8-27B weekday $4/5 \cdot 0/12 \cdot 1/1$ (7 no-answers).

E-obs at the decision point. The E-obs rendering of the 15 clock groups under the decision-point protocols, original $\rightarrow$ E-obs ($\mathrm{both_correct}$, pointwise / pairwise): Qwen3.5-35B-A3B date $1/3 \rightarrow 2/3$ / $1/3 \rightarrow 3/3$, weekday $4/6 \rightarrow 6/6$ / $5/6 \rightarrow 6/6$; Qwen3.8-27B weekday $4/6 \rightarrow 6/6$ / $6/6 \rightarrow 6/6$, date $3/3$ throughout; deepseek-v4.1-flash at ceiling except weekday pointwise ($4/6$). The weekday pairs are observation-free, so their gain under E-obs shows that a verifier can use the clock on a pair that does not require it; the pair label describes the pair and not the verifier's strategy.

Evaluation unit: turn against call on list selection (18 groups). Turn-level / call-level $\mathrm{both_correct}$: deepseek-v4.1-flash and Qwen3.8-27B DP pairwise $7/7$ eligible (11 excluded) / $18/18$; Qwen3.5-35B-A3B DP pointwise / pairwise – / $17/18$ (1 no-answer), $18/18$; ToolRM-Gen $12/18$ (3 ties) / $18/18$; ToolPRM-GRPO v4 $7/18$ (11 no-answer) / $14/18$ (4 no-answer); Skywork-V2 $8/18$ / $16/18$; Agent-RRM $3/18$ (11 ties) / $8/18$ (9 ties); ToolPRM-GRPO v3 $2/18$ / $2/18$ (10 ties, 3 no-answer). With the list withheld from the call-level prefix (`<tool output withheld>`), $\mathrm{both_correct}$ on the 18 groups falls under pointwise scoring from $18/18$, $18/18$, and $17/18$ to $0/18$, $3/18$, and $3/18$ for deepseek-v4.1-flash, Qwen3.8-27B, and Qwen3.5-35B-A3B, and under pairwise comparison from $18/18$ to $5/18$, $6/18$, and $7/18$ (one Qwen3.8-27B cell unresolved). The groups that survive under pairwise comparison all come from three tasks: the airport-code tasks `multi_turn_base_159` and `185`, whose candidate values appear nowhere in the prefix or the instruction, and the technology-sector task `multi_turn_base_112`, whose candidates are symbols

from the benchmark's fixed sector list. The surviving verdicts are therefore consistent with a prior over the benchmark's fixed lists rather than with reading the prefix, and the withheld-list control is informative on the remaining groups.

Detection three ways. Mismatched cells for pointwise judges are split into detected (-1 on the first changed step), missed (0 or $+1$), and no answer. The answered-only miss rate is the blindness measure; the no-answer rate is a decoding measure. On relative-date groups, deepseek-v4.1-flash detects $20/20$ with no no-answers; Qwen3.8-27B detects $8/11$ answered cells with 9 no-answers (reasoning loops of 18–22k characters on time-zone arithmetic); the 35B-A3B mixture judge detects $1/20$ with no no-answers, and 15 of its 19 misses are well-formed $+1$ labels. Labels of 0 ("cannot judge") are counted as abstentions; under minimum aggregation they produce the same selection failure as $+1$.

Figure 11 summarizes these conditions.

Trained verifiers under original / Mask / E-obs / annotation control. $\mathrm{both_correct}$ in native format; t = ties, n = budget-exhausted runs. ToolSandbox weekday (6): ToolRM-Gen 0.17 (5t) / 0.00 (6t) / 1.00 / 0.00 (6t); ToolPRM-GRPO v4 0.00 (2t 4n) / 0.00 (6n) / 1.00 / 0.00 (6n); Skywork 0.00 / 0.00 / 0.00 / 0.00; Agent-RRM@4096 0.00 (4t) / 0.00 (3t) / 0.17 / 0.00; ToolPRM-GRPO v3 0.00 / 0.00 / 0.33 / 0.00. ToolSandbox date (3): ToolRM-Gen 0.33 / 0.33 / 1.00 / 0.33; ToolPRM-GRPO v4 0.33 (1n) / 0.33 (2n) / 1.00 / 0.00; Skywork 0.33 / 0.33 / 1.00 / 0.33; Agent-RRM@4096 0.00 / 0.33 / 0.33 / 0.33. τ^2-bench ISO date (5): ToolRM-Gen 1.00 / 0.00 (3t) / 1.00 / 1.00; ToolPRM-GRPO v4 0.60 (1t 1n) / 0.00 (5n) / 0.80 / 1.00; Skywork 0.60 / 0.00 / 0.80 / 0.60; Agent-RRM 0.00–0.20. On τ^2-bench the annotation-control render is identical to the original one (6 render fields per group; 60/60 prompt pairs identical by hash), so that column is a rerun. Cell-level agreement between the two runs of identical prompts (repeat variance): ToolRM-Gen, Skywork, Qwen3.5-35B-A3B 1.00; Agent-RRM@4096, Qwen3.8-27B, deepseek-v4.1-flash 0.95; ToolPRM-GRPO v4 0.90; ToolPRM-GRPO v3 0.80; Agent-RRM@1200 0.45. ToolPRM-GRPO v4's τ^2-bench 0.60 / 1.00 (one non-terminating run and one tie resolved on rerun) lies inside this variance; the ToolSandbox E-obs recoveries ($0.00 \to 1.00$ on $6/6$) do not. Agent-RRM@1200 disagrees with itself on more than half of identical cells and its condition differences are not interpreted. The renderer now detects no-op conditions by byte comparison of the render fields and labels such columns as reruns. ToolPRM-GRPO v4's no-answers are reasoning loops of 20–38k characters that exhaust the 8k budget; under E-obs the median output falls from $12{,}656$ to $1{,}341$ characters. The Qwen3.5-35B-A3B all-zero rows first obtained for these groups were a harness-step artefact; all numbers here are with harness steps dropped. τ^2-bench step-label rows for judges are excluded (false-alarm saturation from the policy instruction), and τ^2-bench judge evidence is reported under decision-point pairwise only.

Trained models on computed classes. ToolPRM-GRPO v4 fails to produce output on $6/21$ hour groups and $4/18$ list groups, so its class numbers are reported with answered-only companions; ToolRM-Gen answers every group. Agent-RRM is low in every class including hour conversion (0.10) and is therefore insensitive to the argument change at trajectory level rather than sensitive to the relation class; it is excluded from the class comparison.

Decision point at the call. When the listing call and its consumer share one benchmark turn ($11/18$ groups), a turn-level prefix does not contain the observation the selection depends on. Splitting the turn at the call restores the observation to the prefix and makes every list-selection group eligible; the table above gives both turn-level and call-level results. The list-withheld control above confirms that the restored observation is what the verifiers read.

I τ^2-BENCH PREFIXES AND POLICY COMPLIANCE

τ^2-bench gold trajectories are action lists without the authentication call and the confirmation turn that the domain policy requires. The omission is identical in all four cells and for both candidates, so it does not affect which action meets the edited requirement, which is what (C3) and the predicate check. It does affect the full-trajectory step-label read-out: with the policy in the prompt, every judge labels the substituted step -1 in every cell (matched-cell correctness 0.00), because the step violates

the policy whichever instruction is given; τ^2-bench groups are therefore excluded from the step-label tables.

Compliant prefixes. To test whether the decision-point results depend on the omission, we inserted into the shared prefix of every τ^2-bench group, identically in all four cells and before the listing step, the authentication call with its observation from the simulator and a user turn giving the confirmation the policy requires, re-executed all four cells, and re-verified the checker diagonal and the predicate; the candidate step is byte-identical. Predictions recorded before the run: (i) majority outcomes on the 11 selection groups unchanged for every judge, with Qwen3.5-35B-A3B's fixed preference persisting; (ii) pointwise scoring on the compliant prefixes rarely places both candidates at the floor; (iii) ISO-date and baggage outcomes unchanged. Table 6 gives the results. Pairwise, three calls per group: deepseek-v4.1-flash and Qwen3.5-35B-A3B change the majority outcome of none of the 17 groups, while Qwen3.8-27B improves from 1 to 3 of 11 selection groups. Qwen3.5-35B-A3B's fixed preference persists in 27 of 33 selection calls (30 before). Decision-point pointwise on the compliant prefixes solves the ISO-date and baggage groups for every judge and almost none of the selection groups, with 2–10% of rows placing both candidates at the floor.

Table 6: τ^2-bench groups on the original and the policy-compliant prefixes. Pairwise: groups solved by a majority of three calls, original $\rightarrow$ compliant. Pointwise (strict read-out, compliant prefixes only): groups solved, and the share of rows in which both candidates receive the same score at the floor.

	Pairwise, original $\rightarrow$ compliant			Pointwise, compliant			
Judge	select. (11)	ISO date (5)	bag. (1)	select.	ISO date	bag.	floor rows
deepseek-v4.1-flash	$3 \rightarrow 3$	$5 \rightarrow 5$	$1 \rightarrow 1$	0/11	4/5	1/1	7/69
Qwen3.8-27B	$1 \rightarrow 3$	$5 \rightarrow 5$	$1 \rightarrow 1$	2/11	5/5	1/1	2/102
Qwen3.5-35B-A3B	$1 \rightarrow 1$	$5 \rightarrow 5$	$1 \rightarrow 1$	1/11	4/5	1/1	2/96

Prediction (ii) was written on the assumption that the floor concerned decision-point scoring; the original τ^2-bench pointwise rows already had none, so the floor that excludes τ^2-bench is that of the step-label read-out. The selection groups remain unsolved under pointwise scoring for the reason found in §4.2, not because of the omitted steps. τ^2-bench decision-point results are therefore reported under both protocols on the compliant prefixes, and the step-label read-out remains excluded.

J Baselines and availability

Released and used: ToolRM-Gen (Qwen3-4B-Thinking), ToolPRM-GRPO v3 and v4, Web-Shepherd 8B, Skywork-Reward-V2 8B, Qwen2.5-Math-PRM-7B. Not released as of 2026-09-15: ToolRM-Disc, AgentPRM, AgentRM, ToolPRM (Hammer). Data only: the ToolRM training data and fc-reward-bench of Agarwal et al. (2025). Checkpoints used: `RioLee/ToolRM-Gen-Qwen3-4B-Thinking-2507` (Li et al., 2025) and `wjldw/ToolPRM-GRPO-v3`, `-v4` (Lin et al., 2025). ToolPRM-GRPO v4's losses on stated-value groups are reasoning loops with no answer within 8,192 tokens.

K Prompts

All prompted judges receive the prompts below with temperature 0. Each prompt adapts the label scheme and output format of a published design, and the wording is ours: P1 follows the three-way step labels of AgentProcessBench (Fan et al., 2026), P2 the earliest-error format of ProcessBench (Zheng et al., 2024), P3 the three questions of AgentRewardBench (Lù et al., 2025), P4 the promise and progress estimates of AgentPRM (Xi et al., 2026), and P5 the pairwise comparison of ToolPRMBench (Li et al., 2026). P6 is the pointwise counterpart of P5 on the same decision-point input. No prompt contains the reference argument, the checker output, the ground-truth reward, or any step after the decision point. Trained verifiers use the input and output format of their release (Appendix J).

Which prompt each protocol uses. Decision-point pairwise comparison, the primary protocol, uses P5: both slots receive the same shared prefix, followed by one candidate each as a final line `[Candidate] <action>`, and the call is made twice with the candidates in both orders. Full-trajectory pairwise comparison uses P5 with the two complete trajectories. Decision-point pointwise scoring uses P6, one call per candidate. Full-trajectory step labelling uses P1. P2–P4 are secondary protocols.

Slots. {instruction} is the instruction row X_i as the verifier reads it under the condition being run. {tool_schemas} is the benchmark's tool schema list in JSON. {policy_short} is empty for BFCL and ToolSandbox and reads " and the domain policy" for τ^2-bench; {policy_or_constraints} is "the instruction's constraints" or "the domain policy" respectively; {policy_block} is empty unless a section states otherwise (Appendix I). {steps} renders the trajectory or prefix one line per element: a user message as `[User] "<text>"`, and an agent step as `[Step k] role=agent / action: <call> / observation: <output>`, with k counting agent steps from 0.

Pairwise comparison (P5).

```
## Instruction
{instruction}
## Available tools
{tool_schemas}
{policy_block}## Trajectory A
{steps_a}
## Trajectory B
{steps_b}

Considering only the instruction above{policy_short}, is Trajectory A
strictly better than Trajectory B at correctly accomplishing it? Judge
from tool outputs, not from the agents' claims. Answer after brief
reasoning, ending with exactly one line:
Verdict: A | B | TIE
```

Decision-point pointwise scoring (P6). System message:

```
You judge one action taken by a tool-using agent{policy_short}. Score only the
candidate action shown, not the earlier steps, and judge from the instruction and
 the tool outputs
above it rather than from the agent's claims.
```

User message:

```
## Instruction
{instruction}
## Available tools
{tool_schemas}
{policy_block}## Interaction so far
{steps}
## Candidate next action
{action}

How correct is the candidate next action for the instruction above{policy_short}?
 Answer after
brief reasoning, ending with exactly one line:
Score: <integer 0-100>
```

Full-trajectory step labels (P1). System message:

```
You are a strict but fair annotator of tool-using agent trajectories. The
trajectory interleaves user messages, assistant responses, and tool calls
with their outputs. Assign one label to every assistant step, judging it
against the user's instruction{policy_short} and only from the evidence
```

```
shown.

+1  Correct and effective: the step clearly advances task completion,
    e.g., a correct tool call with valid arguments, correctly applying a
    constraint, or correcting an earlier mistake.
 0  Neutral or exploratory: the step is reasonable but has limited or
    unclear effect on progress, e.g., exploratory reasoning, a redundant
    lookup or restatement, a failure caused by the environment rather than
    the agent, or a case where correctness is debatable on the evidence shown.
-1  Incorrect or harmful: the step is factually wrong or counterproductive,
    e.g., misreading the instruction or a tool output, wrong tool or
    arguments, violating {policy_or_constraints}, repeating a failed action
    with no change of strategy, or fabricating tool results or evidence.

Do not credit the agent's own claims of success; rely on tool outputs.
Output JSON only.
```

User message:

```
## Instruction
{instruction}
## Available tools
{tool_schemas}
{policy_block}## Trajectory
{steps}
```

The requested JSON object contains, for every step, the step index, a rationale of at most 400 characters, and a label in $\{-1, 0, 1\}$, together with `first_error_step`, `task_success`, and `success_confidence`.

Earliest error (P2).

```
The following is a task for a tool-using agent and the agent's trajectory.
The agent's steps are enclosed in <step_k> tags and indexed from 0.

[Task]
{instruction}
[Available tools]
{tool_schemas}
{policy_block}[Trajectory]
{steps_tagged}

Review and critique the trajectory step by step. As soon as you identify
an erroneous step, return the index of the step where the earliest error
occurs. If no step is erroneous, return -1. An error is any step that uses
a wrong tool or wrong arguments, contradicts the task{policy_short}, or
relies on information not supported by earlier observations. Put your
final answer (the index) in \boxed{}.
```

Trajectory questions (P3). System message:

```
You are an expert evaluator of tool-using agents. Given the user's goal and
the agent's full interaction history (reasoning, tool calls, tool outputs,
user replies), answer three questions.
```

User message:

```
## Goal
{instruction}
## Available tools
{tool_schemas}
{policy_block}## Interaction history
{steps}
```

```
Answer each question with Yes or No, after brief reasoning.
Q1. Success: Was the sequence of actions successful in achieving the goal?
Q2. Side effects: Did the agent perform unnecessary actions that could lead
    to unintended side effects (e.g., {side_effect_examples})?
Q3. Repetition: Did the agent loop through a sequence of actions that did
    not make progress toward the goal?

Format:
Reasoning: <...>
Success: Yes|No
Side effects: Yes|No
Repetition: Yes|No
```

{side_effect_examples} is "modifying files, or making transactions the goal did not ask for" for BFCL and "changing bookings, orders, or plans the goal did not ask to change" for τ^2-bench.

Prefix promise and progress (P4). System message:

```
You evaluate a partial trajectory of a tool-using agent, truncated after
step {t}. Estimate two quantities, judging only from the evidence shown:
- promise: the probability that the task will ultimately be completed if a
  competent agent continues from the current state. An irreversible wrong
  action already taken should keep this low.
- progress: the fraction of the work required by the instruction that has
  already been correctly accomplished.
Ignore the agent's own claims of success.
Output JSON only:
{"rationale": "<= 3 sentences", "promise": <0-1>, "progress": <0-1>}
```

User message:

```
## Instruction
{instruction}
## Available tools
{tool_schemas}
{policy_block}## Trajectory (steps 1..{t})
{steps_prefix}
```

Condition-specific text. The input conditions of §4 change only the text below; everything else is the prompt above.

- *Caution* (clock and hour groups, Appendix O): the system message "Before judging, check carefully any dates, times, and other values the action depends on."
- *Caution* (selection groups, Appendix Q): the system message "Before judging, check which item the instruction's criterion selects from the listed options."
- *Criterion* (selection groups): the line "Selection criterion: <criterion>" appended to the instruction after a blank line, where <criterion> is one of "lowest price", "highest price", "earliest departure time", "latest departure time", "shortest total travel time", "most recently created", or "earliest created".
- *Literal*: the system message "Restriction for this judgement: decide ONLY by whether the literal values in the candidate's arguments appear in the instruction text. Do not read or use tool outputs, and do not compute, convert, or look up any value."
- *No observation* and *listing withheld*: the affected observations are replaced by <tool output withheld>.
- *No instruction*: the instruction slot and every user turn of the prefix read (instruction withheld).
- *Flat*, *raw table*, and *computed table*: the listing observations are replaced by the renderings described in Appendix Q; the prompt text is unchanged.

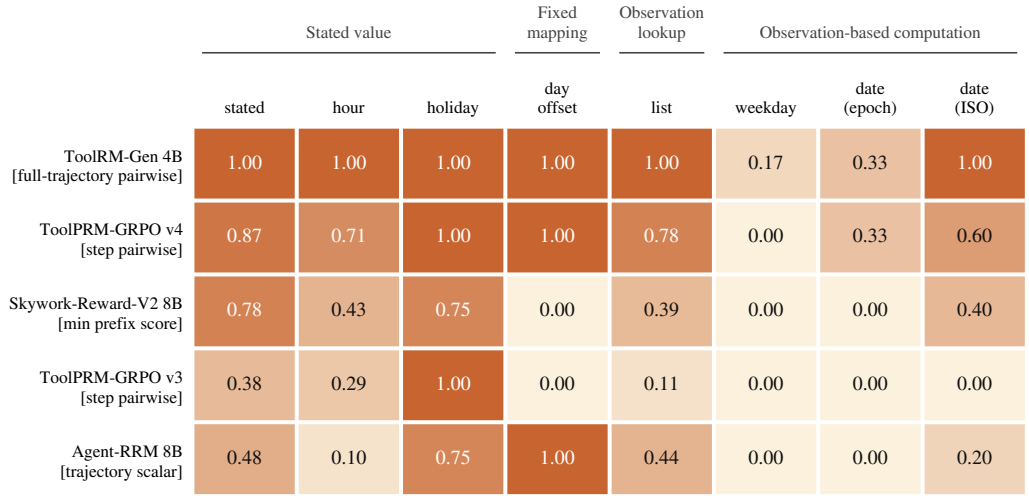

Figure 7: Trained verifiers by relation class in their native format (both_correct; list selection at the call level, other classes at the turn level). Formats are separate blocks and are not ranked against each other. Two formats are at ceiling on stated-value, fixed-mapping, and observation-lookup groups and at 0.00–0.33 on clock computations; Skywork is shown under its minimum-prefix score (Table 19), which ties on many groups, and is at ceiling on the same three relations under its final-prefix score; ToolPRM-GRPO v3 and Agent-RRM are unstable at every class below the stated-value one.

Parsing. A pairwise answer is the last line matching `Verdict: A|B|TIE`; a pointwise answer is the last `Score:` line with a value from 0 to 100. The two orders of a pairwise row are mapped back to candidates, and a row whose two orders disagree is a tie. Under strict parsing, a verdict is read only from a finished answer (finish reason `stop`); otherwise the row is unresolved.

Decoding notes. Qwen3.8-27B under strict JSON-schema decoding produced degenerate outputs (50–60k characters of repeated tabs) in $6/36$ cells and is run with schema enforcement off plus one unconstrained retry; deepseek-v4.1-flash ignores the response-format field and is parsed with a lenient label-only parser whose use is flagged per cell.

L Single-axis rewrites (secondary protocol)

Before the crossed design we scored frozen trajectories under an original instruction X and a single-axis rewrite X' (paraphrase, inversion, scope change, cross-task swap) and reported directional agreement and an anchored magnitude between the paraphrase floor and the swap ceiling. Measured counterfactual labels showed that the naively assumed sign of an inversion is wrong for a plurality of cells, which is the observation that led to requiring execution-graded cells in every group. These results are retained as a secondary analysis.

M Performance across relation classes

Judges by class. Under decision-point pairwise comparison the two large judges are at 1.00 on every class other than the stated values (0.97 for deepseek-v4.1-flash on the 38-group subset) and selection by attribute (Table 10 and Appendix Q). Qwen3.5-35B-A3B fails on two classes under this protocol: dates computed from an epoch timestamp (0.33, $n = 3$, stable over three calls) and selection by attribute (0.33, $n = 15$, stable over three calls), where it keeps the same item under both instructions. The same judge reaches 0.80 on the τ^2-bench groups in which a date is computed from an ISO clock reading ($n = 5$; 1.00 on two further calls), and 0.83 on the weekday pairs, which are observation-free. The epoch and ISO groups come from different benchmarks, so the comparison between them is suggestive only.

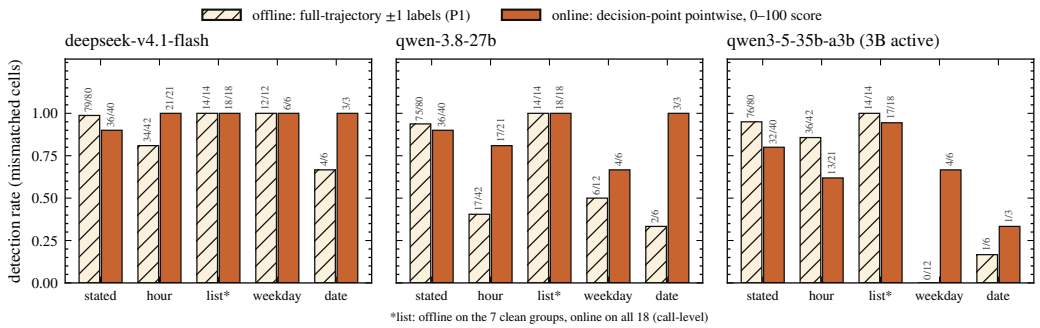

Figure 8: Detection of the perturbed step in mismatched cells for the same verifier and the same decisions under whole-trajectory ± 1 labelling (offline, hatched) and decision-point pointwise scoring (online, filled); bar labels give detected / cells. Detection is not lower offline except for Qwen3.5-35B-A3B on weekday computation; the low whole-trajectory $\mathrm{both_correct}$ values in Table 19 come from matched-cell false alarms and minimum aggregation, and the decision-point pointwise losses from both actions being scored at the floor (§4.4). List selection: offline on the 7 groups whose listing call has its own turn, online on all 18 at the call level.

Trained verifiers by class. Figure 7 reports the five trained verifiers in their native formats. ToolRM-Gen and ToolPRM-GRPO v4 are at or near ceiling on stated-value, fixed-mapping, and list-lookup groups (0.78–1.00 at the call level) and fall to 0.00–0.33 on the ToolSandbox clock families ($n = 6$ and $n = 3$, from one and two scenario families); on the τ^2-bench ISO dates they score 1.00 and 0.60, so neither is defeated by clock-derived dates as such. Skywork-Reward-V2 is shown under the minimum-prefix score of Table 19, under which it loses most groups through ties (the lowest-scoring prefix is the same in both trajectories: 22 of 114 stated-value groups, 11 of 18 list groups); under its final-prefix score it is at ceiling on stated-value, fixed-mapping, and list-lookup groups and at 0.00–0.33 on the clock families. The failing cells are mostly ties and, for ToolPRM-GRPO v4, runs that exhaust the 8k budget in reasoning loops of 20–38k characters, which we count as failures and report separately from abstention. ToolPRM-GRPO v3 and Agent-RRM are low on every class below the stated-value one, including hour conversion (0.29 and 0.10), and are reported as position-unstable and insensitive respectively.

N Whole-trajectory and decision-point detection

Whole-trajectory labelling against decision-point scoring. Figure 8 compares the two regimes for the same verifiers and decisions at the level of detection. Detection is not lower under whole-trajectory labelling: on stated values the three judges detect $79/80$, $75/80$, and $76/80$ perturbed steps against $34/38$, $34/38$, and $30/38$ at the decision point; on hour conversion deepseek-v4.1-flash detects $34/34$ answered cells against $21/21$, and on the clean list-selection groups all three are at $14/14$ against 17–$18/18$. The one place where detection itself differs is Qwen3.5-35B-A3B on the weekday pairs, $0/12$ under whole-trajectory labelling against $4/6$ at the decision point, a single family of six groups. The two regimes also differ in output scale (Qwen3.5-35B-A3B detects $36/42$ hour perturbations with ± 1 labels and $13/21$ with a 0–100 score on the same decisions); runs that vary the input alone and the output alone were not made, so the two factors are not separated here. A low score under whole-trajectory labelling is therefore not by itself evidence that the verifier lacks the decision-point judgment. Appendix I reports why the step-label read-out is not applied to τ^2-bench.

O Input ablation details

Paired input comparisons. In 11 of 18 list-selection groups, a turn-level prefix omits the listing produced earlier in the same turn. Cutting the prefix at the individual call restores the list without revealing the selected action's outcome. Accuracy rises from 0.61 to 1.00 for deepseek-v4.1-flash and improves for all three trained verifiers. Withholding the restored list again reduces pointwise

accuracy from 18, 18, and 17 correct groups to 0, 3, and 3 for the three judges. The benchmark reward cannot detect information omitted only from the verifier input.

The clock experiments isolate a different problem because the original input contains the required information. Rendering an epoch observation as a readable date raises Qwen3.5-35B-A3B from $1/3$ to $3/3$ on the three epoch-date groups under pairwise comparison, while three repeated calls on the original input remain at $1/3$. Full-trajectory detection and several trained verifiers also improve under the readable rendering (Figures 10 and 11). Observation-removal controls show that some weekday pairs can be solved without the clock, while date and list-index pairs deteriorate (Table 17). A targeted set does not support the prediction that the rendering helps more when candidates differ by week than by weekday, and gains also appear on some hour judgments that do not require the clock. The intervention therefore has broader effects than the intended date computation (Appendix R.1 and Appendix M).

Conditions are run with the decision-point runner on the identical 83 groups (BFCL stated-value subset 38, list selection 7 turn-clean groups, ToolSandbox 36). *no observation* replaces every tool output in the prefix by `<tool output withheld>`; *literal* adds to the judge prompt that the action is correct only if its argument appears verbatim in the instruction and that tool output must not be read and nothing computed; *readable* applies the E-obs annotation to every epoch literal in an observation; *rerun* repeats the original input in a fresh call. Rows that errored were re-asked once.

Table 7: Groups solved ($\mathrm{both_correct}$) under decision-point pairwise comparison when the input is altered; n per class. *readable* changes the input only where an observation contains an epoch (all date, weekday, holiday, and fixed-mapping groups; 15 of 21 hour groups); on the remaining groups it is a rerun of the original input. Pair labels as in Table 16. Pointwise results and the rerun noise floor are in Appendix O.

			deepseek-v4.1-flash		Qwen3.5-35B-A3B	
Class	Pair	n	original / no obs. / literal	readable	original / no obs.	readable
Observation-based computation, date	D	3	3 / **1** / **0**	3	1 / 0	**3**
Observation lookup, list index	D	7	7 / **0** / **0**	7	7 / **0**	7
Observation-based computation, weekday	F	6	6 / 6 / **0**	6	5 / 6	6
Fixed mapping	F	2	2 / 2 / **0**	2	2 / 2	2
Stated value, hour	S	21	21 / 21 / **0**	21	21 / 19	21
Stated value, holiday	S	4	4 / 4 / 4	4	4 / 4	4
Stated value	S	38	37 / 36 / 34	36	38 / 34	36

Pointwise results. Groups solved ($\mathrm{both_correct}$, relaxed scoring rule with ties broken at random) under decision-point pointwise scoring, original → no observation / literal / readable: deepseek-v4.1-flash date $3 \to 0/0/3$, list index $7 \to 0/0/7$, weekday $6 \to 6/0/5$, hour $21 \to 21/3/19$, holiday $4 \to 4/4/4$, fixed mapping $2 \to 2/0/2$, stated $34 \to 34/16/35$; Qwen3.5-35B-A3B (no literal run) date $1 \to 0/3$, list index $7 \to 0/7$, weekday $4 \to 0/6$, hour $13 \to 8/20$, holiday $4 \to 4/4$, fixed mapping $2 \to 2/2$, stated $30 \to 23/26$. Under the literal instruction the pointwise judge often scores both actions equally (19 ties on the 40 stated groups).

No instruction. Both rows of a group receive the prefix and the two candidates without any instruction. deepseek-v4.1-flash, pairwise: $0/83$ groups, decided rows 4 of 83 for X_A and 1 of 83 for X_B. Pointwise: $5/83$ groups; X_A row decided in 47 of 83, gold action preferred in 24 of the 47; X_B row decided in 43, gold preferred in 22; the same action chosen in both rows in 15 groups. Per class (pairwise / pointwise groups solved): date $0/0$, list index $0/1$, weekday $0/0$, hour $0/2$, holiday $0/0$, map $0/0$, stated $0/2$.

The zero strict pairwise score is expected for a deterministic instruction-independent preference; the pointwise result with random tie-breaking is outside that guarantee. The decided rows show no strong preference for the original candidate, but this control does not rule out every construction artifact.

Caution and rerun. Prompt for caution: the original prompt followed by "Before judging, check carefully any dates, times, and other values the action depends on." Groups solved at the original input → readable / caution / rerun on the 36 ToolSandbox clock and hour groups: Qwen3.5-35B-A3B pointwise $24 \to 35/30/29$, pairwise $33 \to 36/33/34$; Qwen3.8-27B pointwise $29 \to 36/35/33$, pairwise

Table 8: Share of decided rows whose preference changes when the requirement is edited (crossed), when only the wording is paraphrased, or when the identical input is called again (rerun); 77 groups. Paired counts are rows changed by the paraphrase only against rows changed by the rerun only.

Judge	Protocol	crossed	paraphrase	rerun	paraphrase only : rerun only
deepseek-v4.1-flash	pointwise	0.97	0.02	0.02	0 : 1
deepseek-v4.1-flash	pairwise	1.00	0.01	0.01	0 : 1
Qwen3.8-27B	pointwise	0.93	0.03	0.04	1 : 2
Qwen3.8-27B	pairwise	1.00	0.00	0.00	0 : 0
Qwen3.5-35B-A3B	pointwise	0.94	0.08	0.06	5 : 6
Qwen3.5-35B-A3B	pairwise	1.00	0.00	0.00	0 : 0

$36 \to 36/31/36$; deepseek-v4.1-flash pointwise $36 \to 33/36/36$, pairwise $36 \to 36/36/36$. Of Qwen3.5-35B-A3B's 12 pointwise failures at the original input, 7 fail again on the rerun. Unresolved answers after one retry: Qwen3.5-35B-A3B 0, Qwen3.8-27B 3 (pairwise, caution).

Paraphrase invariance. Both instructions of each group were paraphrased by gpt-5.4-mini under the constraints that every value, name, and placeholder is kept and that the wording changes; 6 of 83 groups had no paraphrase that passed the checks. Median normalised edit distance 0.17, maximum 0.40. Flip rates on decided rows (paraphrase / rerun): deepseek-v4.1-flash pointwise $3/149$ / $3/146$, pairwise $1/152$ / $1/151$; Qwen3.8-27B pointwise $5/143$ / $6/146$, pairwise 0 / 0; Qwen3.5-35B-A3B pointwise $10/133$ / $9/139$, pairwise 0 / 0. Nearly all Qwen3.5-35B-A3B flips are on the hour groups (paraphrase $7/38$, rerun $4/36$). Split at the median edit distance, weak and strong paraphrases flip at the same rate as their reruns (Qwen3.5-35B-A3B $5/65$ against $6/67$ and $5/68$ against $3/72$; Qwen3.8-27B $3/71$ against $4/74$ and $2/72$ against $2/72$; deepseek-v4.1-flash $1/72$ against $1/71$ and $2/77$ against $2/75$).

Paraphrase effects are comparable to repeated-input variation. We paraphrased both instructions of 77 groups while preserving values and placeholders (median normalized edit distance 0.17). On decided rows, changing the requirement reverses the preference in 93–100% of cases, whereas paraphrasing reverses it in 0–8% (Table 8). Paired tests detect no difference between paraphrase and repeated-input reversal rates (McNemar $p = 1.0$). This supports stability for the tested paraphrases; it does not establish invariance to arbitrary wording changes.

Literal-only prompting. When instructed to use only literal matching, deepseek-v4.1-flash retains holidays ($4/4$) and most stated values ($34/38$), but ties on the remaining classes. These include hour conversion, where the instruction states 5 PM and the argument contains 17. This is a prompted restriction, not a guarantee that the model performs no computation.

Reasoning coding. Each pointwise row's reasoning was coded by an LLM given the edited argument and the ground truth for: whether the differing argument is named, whether the violation is explained correctly, and, for rows with a wrong or tied verdict, the cause (both actions at the floor; a value shared by both candidates rejected; a deduction unrelated to the edit; the judge's own error on the required value; other). The shared-value category was added after a first pass filed those rows under unrelated deductions, and all rows were re-coded. Recognised rows: deepseek-v4.1-flash $164/166$, Qwen3.8-27B $156/162$, Qwen3.5-35B-A3B $155/162$. Recognised rows with a wrong or tied verdict: deepseek-v4.1-flash 4 (floor 2, unrelated 1, other 1), Qwen3.8-27B 5 (shared value 2, unrelated 2, own error 1), Qwen3.5-35B-A3B 18 (floor 7, shared value 7, unrelated 3, own error 1). Several Qwen3.5-35B-A3B reasonings are degenerate loops whose final score is arbitrary.

Noise floor. On groups where *readable* leaves the input byte-identical, the column is a rerun. Per-group flips on those groups (up/down): stated pointwise Qwen3.5-35B-A3B $+1/-5$, deepseek-v4.1-flash $+4/-3$; stated pairwise Qwen3.5-35B-A3B $0/-2$, deepseek-v4.1-flash $0/-1$. A separate rerun of the identical input moved Qwen3.5-35B-A3B from 30 to 26 of 38 stated groups under pointwise scoring. Condition effects are read against this floor; the hour gain of Qwen3.5-35B-A3B is $+7/-1$ on the 15 groups whose input changed and $+1/-0$ on the 6 whose input did not.

Reasoning budget. Of Qwen3.8-27B's 27 full-trajectory step-label cells with no parseable label at the 8k budget (18 on hour, 9 on date and weekday groups), a 16k budget resolves 13, all as detections, and leaves 14 unresolved; 32k resolves 18 (16 detected, 2 missed) and leaves 9; with reasoning disabled every cell resolves but only 4 are detections. A larger budget converts unresolved cells into detections slowly, and removing reasoning converts them into misses.

Repeated calls on small classes. The classes with few groups were scored three times with byte-identical input for Qwen3.5-35B-A3B and Qwen3.8-27B, and the τ^2-bench baggage group also for deepseek-v4.1-flash (call 1 is the reported run; call 2 the rerun above; call 3 a further call; for the τ^2-bench selection-by-attribute and baggage groups call 1 is the first judge call on them). Table 9 gives the class accuracy in each call, the share of groups solved in at least two of three calls, and the groups that fail in all three. Every reported value lies inside the range of the three calls. Across the 25 pairwise groups and three calls, the v2 taxonomy gives Qwen3.5-35B-A3B 55 both-correct, 9 fixed, 6 position, 1 partial-tie, and 4 no-answer outcomes; Qwen3.8-27B gives 62, 0, 5, 8, and 0. Qwen3.8-27B solves every τ^2-bench date group in all three calls. The v2 categories also separate failure forms that the per-call accuracy hides: Qwen3.5-35B-A3B's pairwise failures on the epoch-date groups are position-following (5 of 9 group-calls, CER 0.67), and two cells with majority 1.00 still contain committed errors (Qwen3.8-27B pairwise epoch date, CER 0.22; Qwen3.8-27B pointwise weekday, CER 0.06). Pointwise rows have no position category, so their CER counts wrong selections and partial ties whose resolved row is wrong, and is not directly comparable with pairwise CER. Stable failure remains Qwen3.5-35B-A3B on the epoch-date groups, where two of three groups fail in every call under pairwise comparison, and on the selection-by-attribute groups reported in Table 10. deepseek-v4.1-flash's baggage group passes in every call. Conversely, Qwen3.8-27B's weekday value of 0.67 under pointwise scoring (§4.4) is a single-call value with majority 1.00, so the pointwise-to-pairwise gain on that class for Qwen3.8-27B lies within call variation (the hour groups, on which the larger gains occur, were not part of the repeat set).

Table 9: Three byte-identical decision-point calls on the small classes ($\mathrm{both_correct}$ per call). Majority is the share of groups solved in at least two calls; stable counts groups failing in all three. τ^2-bench groups were scored pairwise on the original prefixes (Appendix I gives pointwise results on the policy-compliant prefixes); the selection-by-attribute groups are in Table 10.

Judge	Protocol	Class	n	call 1 / 2 / 3	majority	stable
deepseek-v4.1-flash	pairwise	Observation-based computation, baggage (τ^2-bench)	1	1.00 / 1.00 / 1.00	1.00	0
Qwen3.5-35B-A3B	pointwise	Observation-based computation, date (epoch)	3	0.33 / 0.33 / 1.00	0.33	0
Qwen3.5-35B-A3B	pointwise	Observation-based computation, weekday	6	0.67 / 0.50 / 0.67	0.50	1
Qwen3.5-35B-A3B	pointwise	Stated value, holiday	4	1.00 / 1.00 / 1.00	1.00	0
Qwen3.5-35B-A3B	pointwise	Fixed mapping	2	1.00 / 1.00 / 1.00	1.00	0
Qwen3.5-35B-A3B	pairwise	Observation-based computation, date (epoch)	3	0.33 / 0.33 / 0.33	0.33	2
Qwen3.5-35B-A3B	pairwise	Observation-based computation, weekday	6	0.83 / 1.00 / 1.00	1.00	0
Qwen3.5-35B-A3B	pairwise	Stated value, holiday	4	1.00 / 1.00 / 1.00	1.00	0
Qwen3.5-35B-A3B	pairwise	Fixed mapping	2	1.00 / 1.00 / 1.00	1.00	0
Qwen3.5-35B-A3B	pairwise	Observation-based computation, date (ISO, τ^2-bench)	5	0.80 / 1.00 / 1.00	1.00	0
Qwen3.5-35B-A3B	pairwise	Observation-based computation, baggage (τ^2-bench)	1	1.00 / 1.00 / 1.00	1.00	0
Qwen3.8-27B	pointwise	Observation-based computation, date (epoch)	3	1.00 / 1.00 / 0.67	1.00	0
Qwen3.8-27B	pointwise	Observation-based computation, weekday	6	0.67 / 1.00 / 0.83	1.00	0
Qwen3.8-27B	pointwise	Stated value, holiday	4	0.75 / 1.00 / 1.00	1.00	0
Qwen3.8-27B	pointwise	Fixed mapping	2	1.00 / 1.00 / 1.00	1.00	0
Qwen3.8-27B	pairwise	Observation-based computation, date (epoch)	3	1.00 / 0.67 / 0.67	0.67	0
Qwen3.8-27B	pairwise	Observation-based computation, weekday	6	1.00 / 1.00 / 1.00	1.00	0
Qwen3.8-27B	pairwise	Stated value, holiday	4	1.00 / 1.00 / 1.00	1.00	0
Qwen3.8-27B	pairwise	Fixed mapping	2	1.00 / 1.00 / 1.00	1.00	0
Qwen3.8-27B	pairwise	Observation-based computation, date (ISO, τ^2-bench)	5	1.00 / 1.00 / 1.00	1.00	0
Qwen3.8-27B	pairwise	Observation-based computation, baggage (τ^2-bench)	1	1.00 / 1.00 / 1.00	1.00	0

P SINGLE-ROW ACCURACY AGAINST THE CROSSED SCORE

For each judge, the original-instruction row (X_A) is scored as a conventional preference benchmark would score it, and the crossed score is compared with it. An additional failure is a group whose X_A row is correct and whose X_B row fails. On the 15 selection groups in the first pairwise call, the five judges answer 42 judge–group pairs correctly under X_A, of which 12 fail under X_B: seven

fixed preferences, two position-following outcomes, one tie, and two unresolved outputs. The judge-level original / crossed counts are deepseek-v4.1-flash $10/7$, Qwen3.8-27B $7/6$, Qwen3.5-35B-A3B $8/5$, GPT-5.6 Luna $6/5$, and GPT-5.6 Terra $11/7$. For the original three judges, every class outside selection by attribute has zero additional failures in this run. Pooled over all 113 groups, their original / crossed / additional-failure rates are deepseek-v4.1-flash $0.95/0.92/0.03$, Qwen3.8-27B $0.93/0.91/0.02$, and Qwen3.5-35B-A3B $0.91/0.88/0.04$. Most of Qwen3.5-35B-A3B's fixed preferences on the selection groups occur where the X_A row is already wrong; a single-row benchmark records these only as ordinary errors.

Q SELECTION-BY-CRITERION GROUPS

Table 10: Selection-by-attribute results on 15 groups with three calls per group. Tie combines explicit ties and partial ties; CER counts fixed, inverted, position-following, and partial-tie outcomes whose resolved row is wrong.

Judge	Solved	Both	Fixed	Inverted	Position	Tie	Unresolved	CER
deepseek-v4.1-flash	7	21	5	4	8	1	6	.40
Qwen3.8-27B	5	17	6	0	14	3	5	.47
Qwen3.5-35B-A3B	5	15	28	0	1	1	0	.64
GPT-5.6 Luna	5	15	3	0	3	24	0	.18
GPT-5.6 Terra	7	21	1	0	4	19	0	.11

Construction. Each group edits only the phrase that names the selection criterion; every other value, name, and placeholder is kept. The target of the edited call is the item that the criterion picks from the listing observation in the prefix, recomputed by an independent predicate (argmin or argmax over price, departure time, duration, or reminder time; a stated time bound such as "not before 11am" is applied as a filter). Groups are rejected when two observed items tie under a criterion in a way that lets both candidates satisfy the edited instruction, when only one candidate remains after the instruction's restrictions, when a candidate trajectory fails to execute, or when the task's graded assertions fix an amount that only one candidate can produce. No candidate identifier occurs in any instruction, so the string-matching scorer solves none of the groups; the observation-blind predicate fails on all of them. The 15 groups are τ^2-bench airline tasks 15, 16, and 20 (nine groups; the itineraries of tasks 15 and 16 are shared, so they form one cluster), τ^2-bench retail tasks 49 and 56 (two variants and seven variants respectively), and one ToolSandbox reminder scenario with its three tool-augmented variants (one cluster).

Runs and settings. The original condition uses decision-point pairwise comparison in both orders with three calls per group and judge. The listing-withheld condition replaces the search or listing observation by `<tool output withheld>`; the original three judges have one call on the diagnosed set, while the GPT judges have three. No judge solves a group in this condition. The non-GPT original runs use an 8,192-token output budget with the recorded retry and lenient parsing rules. GPT-5.6 Luna and Terra use a 2,560-token budget and strict parsing in every condition; reasoning tokens count toward this limit, and no generation reaches it. Each released row records the routed model identifier and provider.

Input changes. *Criterion* appends one line that restates the named selection criterion. *Caution* asks the judge to check which listed item satisfies that criterion. *Flat* mechanically places every leaf object and field in rows. *Raw table* groups legs by itinerary and retains only observed values for the booked cabin. *Computed table* adds derived item attributes such as total price and duration. The candidate calls and every other observation remain unchanged. Criterion restatement and caution do not consistently improve the original three judges. Computed tables yield CER $= 0$ for four of five judges on both group sets; GPT-5.6 Luna has one position-following call on each set (CER 0.02 and 0.03). Raw tables yield CER $= 0$ on the diagnosed groups; on held-out they leave committed errors for deepseek-v4.1-flash (CER 0.12) and none for the GPT judges. The renderer comparison and its two-factor controls are reported below.

`[Step 0] search_direct_flight(origin=ATL, destination=PHL, date=2024-05-24)` → (no itineraries)
`[Step 1] search_onestop_flight(origin=ATL, destination=PHL, date=2024-05-24)` →

itinerary	stops	departs	arrives	duration	price (economy)	status
HAT110 + HAT132	1	14:00	18:00	4h00m	295	available
HAT110 + HAT172	1	14:00	00:00+1	10h00m	207	available
HAT110 + HAT206	1	14:00	19:00	5h00m	250	available
HAT110 + HAT264	1	14:00	00:00+1	10h00m	261	available
HAT164 + HAT132	1	11:00	18:00	7h00m	335	available
HAT164 + HAT172	1	11:00	00:00+1	13h00m	247	available
HAT164 + HAT206	1	11:00	19:00	8h00m	290	available
HAT164 + HAT264	1	11:00	00:00+1	13h00m	301	available
HAT227 + HAT139	1	11:00	19:00	8h00m	216	available
HAT227 + HAT271	1	11:00	21:00	10h00m	250	available

`[Candidate A] update_reservation_flights(reservation_id=M05KNL, cabin=economy, flights=[HAT110, HAT172], ...)`
`[Candidate B] update_reservation_flights(reservation_id=M05KNL, cabin=economy, flights=[HAT164, HAT132], ...)`

Figure 9: The table condition for one selection group (τ^2-bench airline task 15, cheapest → earliest departing). Only the two search observations change: the raw search result (one JSON object per leg with per-cabin prices and seat counts) is replaced by one row per itinerary with the attributes the criteria use, in the search result's order and not sorted by any criterion. The instruction and both candidate calls are those of the original input. Candidate A is the cheapest itinerary (207) and candidate B one of the six that tie for the earliest departure (11:00); picking the item, and noticing the tie, is left to the judge.

Held-out groups. Eleven groups on six previously unused listings comprise four τ^2-bench airline reservations and two τ^2-bench retail orders, with the same criterion pairs and validation checks. Three calls at the original input give majority scores of $2/11$, $2/11$, $1/11$, $1/11$, and $1/11$ for deepseek-v4.1-flash, Qwen3.8-27B, Qwen3.5-35B-A3B, GPT-5.6 Luna, and GPT-5.6 Terra. The Qwen3.8-27B original-input value in Table 12 comes from a separate equal-budget rerun (Appendix Q), which gives $1/11$; the three-call run reported here gives $2/11$. Of 33 group-calls, deepseek-v4.1-flash and Qwen3.8-27B each leave 18 unresolved, Qwen3.5-35B-A3B makes 10 fixed-preference selections, and the GPT judges predominantly tie. With the listing withheld, every judge scores $0/11$; all 11 Qwen3.8-27B calls are ties, eight explicit and three mixed. The previously unclassified 11 Qwen3.5-35B-A3B original-input calls are unresolved truncations. These groups are not part of the 195-group inventory.

Renderer controls. We compare the original listing with three renderings using three calls per group. *Flat* is task-agnostic: it places each leaf object in one row, preserves every field and the original order, and computes no value. *Raw table* places one itinerary in each row and retains the per-leg departure, arrival, and booked-cabin price, again without computing totals. *Computed table* places one item in each row and includes derived attributes used by the selection criteria. The last two renderings therefore expose item boundaries; raw table also reads the booked cabin from the candidates, while computed table uses the criterion to select its columns. The four-rendering comparison covers deepseek-v4.1-flash and the GPT-5.6 judges; Table 12 reports original versus computed-table results for all five judges.

Flat reaches $6/6$ group-calls on the diagnosed retail groups for all three judges, where each candidate is a single object. Its gains are much smaller on the multi-leg airline groups, which account for the gap between flat and raw table. We isolate two differences on the 9 diagnosed and 9 held-out airline groups: whether rows represent legs or complete itineraries, and whether prices for every cabin or only the booked cabin are retained. On the diagnosed groups, either change alone is often sufficient. On the held-out groups, the combination is strongest: once legs are grouped into itinerary rows, removing other cabins' prices recovers 4–12 additional group-calls across the three judges; removing them while retaining one row per leg recovers none. These controls show that the recovery

Table 11: Selection groups solved by a majority of three calls under the renderer controls. GPT-5.6 Luna's original-input column comes from an earlier run of the same configuration; solved counts agree across repeated runs.

Set	Judge	Original	Flat	Raw table	Computed table
Diagnosed (15)	deepseek-v4.1-flash	7	9	15	15
	GPT-5.6 Luna	5	9	12	14
	GPT-5.6 Terra	7	9	15	15
Held-out (11)	deepseek-v4.1-flash	2	3	10	11
	GPT-5.6 Luna	1	2	5	8
	GPT-5.6 Terra	1	3	8	11

Table 12: Original listing versus computed table for all five judges. Each cell reports groups solved by a majority of three calls, with CER in parentheses. The non-GPT original runs use an 8,192-token output budget, except the held-out original values of deepseek-v4.1-flash and Qwen3.8-27B, which come from the separate equal-budget rerun at 16,384 tokens (Appendix Q); all non-GPT table runs use 16,384 tokens. GPT-5.6 runs use 2,560 tokens in both conditions. The equal-budget control in Appendix Q reproduces the deepseek-v4.1-flash held-out recovery. †The 11 previously missing Qwen3.5-35B-A3B held-out calls are classified as unresolved truncations.

	Diagnosed (15)		Held-out (11)	
Judge	Original	Table	Original	Table
deepseek-v4.1-flash	7 (.40)	15 (.00)	2 (.27)	11 (.00)
Qwen3.8-27B	5 (.47)	15 (.00)	1 (.21)	11 (.00)
Qwen3.5-35B-A3B	5 (.64)	15 (.00)	1 (.42)	11† (.00)
GPT-5.6 Luna	5 (.18)	14 (.02)	1 (.09)	8 (.03)
GPT-5.6 Terra	7 (.11)	15 (.00)	1 (.00)	11 (.00)

depends on presenting the compared items and their relevant observed attributes together, rather than on converting JSON to an arbitrary table.

Equal-budget control. The original-input and table runs above used different output-budget settings. To control for this difference, we evaluate deepseek-v4.1-flash on the same 11 held-out groups with a 16,384-token output limit for both conditions, using Fireworks, temperature zero, strict verdict parsing, and the same retry rule (up to two requests per instruction and presentation order). Each group is evaluated three times in both candidate orders under both instructions. The original listing yields $2/11$ solved groups in each repetition; the computed table yields $11/11$ in each, giving majority scores of $2/11$ and $11/11$. Across the 33 paired group-calls, the table recovers 27 and breaks none. No empty provider responses remain in this comparison. Original-input generations reach the output limit in 8, 6, and 12 of 44 generations across the three repetitions, compared with none under the table. Mean completion-token counts per generation are approximately 9.7k, 8.3k, and 9.4k for the original listing and 1.2k, 0.9k, and 1.1k for the table. Thus, the held-out recovery for this judge persists when the allowed output budget is matched. The same rerun for Qwen3.8-27B gives $1/11$ for the original listing and $11/11$ for the table in each repetition; the table recovers 30 of 33 paired group-calls and breaks none, and 8 of 132 original-input generations returned empty provider responses.

Reading the failures. The ordering by difficulty follows the number of items and the arithmetic the criterion requires: reminder timestamps (four groups, every judge $3/3$), two retail variants ($3/3$ for every judge), seven retail variants (no judge passes), multi-leg itineraries whose price and duration are sums over legs (nearly no passes). Qwen3.5-35B-A3B's fixed item in the airline groups is never the itinerary that ties for the earliest departure, and it is the fastest itinerary when the pair is cheapest against fastest, so the fixed choice is a property of the group's items rather than of a criterion; the table condition above shows that it disappears once the items' attributes are computed. The remaining failure forms include Qwen3.8-27B's position-following, deepseek-v4.1-flash's truncated reasoning

on the airline groups, where 8–12 of 44 generations reach the 16,384-token limit in the equal-budget runs, and deepseek-v4.1-flash's inverted preference on the earliest-to-latest pairs.

Results by listing. Table 13 aggregates the selection results by independent listing, the observed search result or order on which a set of groups is built. On the diagnosed set, the crossed failures are concentrated in the airline listings and in the retail order with seven variants: the eight groups on reservation M05KNL are solved by majority in 3 of 40 judge–group pairs, whereas the four reminder groups and the retail order with two variants are solved by every judge. On the held-out set, only the retail order #W1006327 is solved by every judge. Computed tables are near ceiling across listings (74/75 diagnosed and 52/55 held-out judge–group pairs). The renderer recovery is not driven by a single listing: computed tables improve or preserve performance on every diagnosed and held-out listing.

Table 13: Selection results by listing. Each entry counts judge–group pairs solved by a majority of three calls, pooled over the five judges; the denominator is five times the number of groups. X_A scores only the original instruction, Crossed requires both instructions, and Table uses the computed-table rendering. Original-input runs are those of Table 10 and the held-out paragraph above.

Set	Listing	Groups	X_A	Crossed	Table
Diagnosed	τ^2-bench airline, reservation M05KNL	8	10/40	3/40	40/40
	τ^2-bench airline, task 20	1	5/5	1/5	4/5
	τ^2-bench retail, order #W3470184	1	5/5	5/5	5/5
	τ^2-bench retail, order #W4284542	1	3/5	0/5	5/5
	ToolSandbox reminders	4	20/20	20/20	20/20
	Total	15	43/75	29/75	74/75
Held-out	τ^2-bench airline, reservation 672VDE	2	2/10	0/10	10/10
	τ^2-bench airline, reservation LJM636	3	4/15	2/15	14/15
	τ^2-bench airline, reservation P22235	3	0/15	0/15	13/15
	τ^2-bench airline, reservation DGLJTR	1	2/5	0/5	5/5
	τ^2-bench retail, order #W1006327	1	5/5	5/5	5/5
	τ^2-bench retail, order #W1067251	1	0/5	0/5	5/5
	Total	11	13/55	7/55	52/55

Second selection type pilot. We construct four BFCL TradingBot groups in two clusters. The prefix calls `get_stock_info` separately for five symbols, and the edited criteria select the highest against lowest price or the stock furthest above against below its 20-day moving average. All four groups pass the checker diagonal, predicate reversal, observation-blind, symbol-leak, no-tie, and position checks. With the observations present, deepseek-v4.1-flash, Qwen3.5-35B-A3B, and Qwen3.8-27B solve $4/4$ groups, while GPT-5.6 Luna and Terra solve $3/4$. With all five stock observations withheld, every judge scores $0/4$. This pilot establishes feasibility on a second selection type whose evidence is spread across tool calls, but its four near-ceiling groups do not test whether the original failure or renderer recovery transfers.

R SELECTION ON HELD-OUT ACTIONS

Overview and results. We test two effects on 208 policy-generated decision points where at least one candidate succeeds and another fails after a fixed continuation. The pool contains 67 held-out clock-family points, 58 sibling-family points, and 83 clock-free controls. These labels measure completed-task success under the continuation rather than local action correctness. The other two diagnoses cannot be tested on this pool, which contains no listing-then-selection decisions or full-trajectory scores. The construction and predictions recorded before the held-out verifier results were inspected are detailed below.

Readable clock observations improve selected-task success for both prompted judges. On time-value points, the change is $+0.08$ to $+0.10$ for deepseek-v4.1-flash and $+0.13$ for Qwen3.5-35B-A3B, but caution helps Qwen3.5-35B-A3B by a comparable amount and the trained verifiers show no clear gain. The representation effect therefore transfers to some decisions but is not verifier-specific.

Table 14: Selection on policy-generated candidates at clock decision points where candidates differ in a time value, on points with a resolved verdict. Base is selected-task success under the fixed continuation; changes are relative to base. Brackets give 95% intervals over scenarios. Answer rates and the failure-inclusive accounting are in Appendix R.

Verifier	Protocol	Base	Δ render	Δ caution	Δ rerun
deepseek-v4.1-flash	DP pointwise	0.63	$+0.08$ $[-0.07, +0.16]$	-0.06	$+0.01$
deepseek-v4.1-flash	DP pairwise	0.80	$+0.10$ $[+0.01, +0.19]$	$+0.00$	-0.01
Qwen3.5-35B-A3B	DP pointwise	0.54	$+0.13$ $[+0.03, +0.23]$	$+0.15$ $[+0.00, +0.32]$	$+0.04$ $[-0.12, +0.15]$
ToolRM-Gen	pairwise	0.64	$+0.00$ $[-0.12, +0.19]$	–	–
ToolPRM-GRPO v4	pairwise	0.52	$+0.04$ $[-0.07, +0.15]$	–	–

The scoring effect is more concentrated and depends on how unresolved verdicts are counted. On points where both protocols return a verdict, deepseek-v4.1-flash is unchanged over all points and Qwen3.8-27B gains $+0.06$ with an interval that includes zero; on clock points Qwen3.8-27B gains $+0.09$ [$+0.01$, $+0.17$] against $+0.01$ on controls, and on time-value points deepseek-v4.1-flash gains $+0.12$ [$+0.00$, $+0.26$]. The difference between the time-value and control effects has an interval that includes zero for every judge, so the pattern is consistent with the crossed diagnosis rather than a confirmation of it. Pairwise comparison also returns fewer verdicts: Qwen3.5-35B-A3B resolves 39% of points under pairwise comparison against 90% under pointwise scoring, and once an unresolved point counts as a failure its apparent pairwise gain becomes a loss of -0.18 [-0.28, -0.09]; deepseek-v4.1-flash's time-value gain becomes -0.11 under the same accounting (answer rates 0.86 against 0.95), and Qwen3.8-27B, which resolves 97%, keeps $+0.03$. The representation effect for deepseek-v4.1-flash under pointwise scoring ($+0.09$ [$+0.05$, $+0.11$] on clock points) holds under every accounting. Direct comparison therefore recovers judgments that a judge completes, and completing them is part of the outcome; the representation effect transfers without predicting which verifier improves.

Design. Held-out pool: ToolSandbox scenario families that appear in no crossed group (clock families `find_thanksgiving_timestamp`, `remove_reminder_with_recency_latest`, `search_reminder_with_recency_upcoming`, `search_reminder_with_recency_yesterday`; two sibling families that share a relation with a crossed family; ten clock-free control families), each with augmented variants (distraction tools, scrambled names and descriptions). Decision points are every tool-call step of reference rollouts from three policies; candidates are the actions that eight policies propose at that prefix plus the reference action, merged by function and arguments, at most five per point. Each candidate is executed and a fixed reference policy completes the task; success is milestone similarity ≥ 0.85, a threshold fixed from the outcome distribution before any verifier result was read. The pool has 1,017 points with at least two distinct candidates and 3,218 executed candidates, of which 208 points are separating (at least one candidate succeeds and one fails). Conditions: *base*; *render*, every epoch in a prefix observation annotated with its date and weekday (candidates untouched); *caution*, base plus "Before judging, check carefully any dates, times, and other values the action depends on"; *rerun*, base in a fresh call. Verifiers: the three judges under pointwise and pairwise decision-point protocols, ToolRM-Gen and ToolPRM-GRPO v4 in their own formats (base and render). Effects are within-verifier paired differences on the same points; intervals are bootstrapped over base scenarios.

Amendments and interpretation. The initial scored subset was expanded to separating points after inspecting candidate outcomes, before the API judges' held-out results were read. Prefixes from two weaker policies were also added. Repeated-input calls were introduced after variability was observed on crossed groups. The variant annotating candidate actions as well as observations was added after inspecting trained-verifier results and is exploratory. These changes are distinguished from the original predictions rather than treated as an unchanged evaluation plan. The reported selection labels concern success under a fixed continuation; recovery or later failure can separate

them from local action correctness. Clock results that combine held-out and sibling families are labelled as combined results. Effects on separating points do not estimate the intervention's average effect over all decision points.

Predictions recorded before reading held-out results and outcomes. P1, Qwen3.5-35B-A3B selects worse than deepseek-v4.1-flash on clock points with a larger gap than on control points: direction as predicted (0.48 against 0.60 on clock, 0.36 against 0.44 on control), the clock gap is only slightly larger and the proposed class-specific difference is not established. P2, render helps Qwen3.5-35B-A3B and the trained verifiers and not deepseek-v4.1-flash: Qwen3.5-35B-A3B improves ($+0.13$ [$+0.03$, $+0.23$] on time-value points, rerun $+0.04$), deepseek-v4.1-flash improves by a similar amount ($+0.08$ to $+0.10$), the trained verifiers do not move; partly failed. P3, render is neutral on control points: holds for deepseek-v4.1-flash and the trained verifiers; Qwen3.5-35B-A3B worsens (-0.16, $n = 16$); failed. P4, caution helps less than render: holds for deepseek-v4.1-flash; fails for Qwen3.5-35B-A3B ($+0.12$ to $+0.15$). An exploratory variant that also annotates epochs inside the candidates is no better than render.

The comparison in Table 14 reports selection on the time-value subset; the design and amendments above specify its scope.

Coverage-inclusive accounting. Answer rates over the 208 points (pointwise / pairwise): deepseek-v4.1-flash 0.95 / 0.86, Qwen3.8-27B 1.00 / 0.97, Qwen3.5-35B-A3B 0.90 / 0.39. Pairwise minus pointwise selection success over all points, on resolved points only / with an unresolved point counted as a failure / with an unresolved point counted as a random pick: deepseek-v4.1-flash -0.00 / -0.07 [-0.16, $+0.04$] / -0.01; Qwen3.8-27B $+0.06$ / $+0.03$ / $+0.05$; Qwen3.5-35B-A3B $+0.07$ / -0.18 [-0.28, -0.09] / $+0.06$. On time-value points the failure-inclusive accounting turns deepseek-v4.1-flash's $+0.12$ into -0.11 and Qwen3.5-35B-A3B's gain into -0.34. The difference between the time-value and control effects, $\Delta_{\text{time}} - \Delta_{\text{control}}$, has a bootstrap interval that includes 0 for every judge under every accounting; only on resolved points is its point estimate positive ($+0.10$ deepseek-v4.1-flash, $+0.17$ Qwen3.5-35B-A3B). For the render condition, deepseek-v4.1-flash's pointwise gain on clock points ($+0.09$ [$+0.05$, $+0.11$]) and its difference from the control points hold under all three accountings; deepseek-v4.1-flash's pairwise render gain is significant only on resolved points, because its pairwise answer rate falls from 0.86 to 0.71 under render, and the effect disappears once unresolved points count ($+0.06 \rightarrow -0.03$ on clock points).

Pointwise against pairwise on the same points. For each separating point and judge, the candidate picked under pointwise scoring (highest score; ties share credit) and under pairwise comparison (both orders) over the identical candidate set; success is the picked candidate's completed-task success. Paired points are those with a resolved pick under both protocols: deepseek-v4.1-flash 172 of 208 (pairwise resolved on 178), Qwen3.8-27B 202 (pairwise resolved on 202, after retries), Qwen3.5-35B-A3B 79 (pairwise resolved on 80). Differences are pairwise minus pointwise with intervals bootstrapped over base scenarios; $b : c$ counts points that only pairwise or only pointwise picks correctly. deepseek-v4.1-flash: all -0.00 [-0.09, $+0.11$] (22:23); clock families $+0.07$ [-0.18, $+0.25$]; sibling families -0.11; control $+0.01$; time-value $+0.12$ [$+0.00$, $+0.26$] (5:0, exact $p = 0.06$). Qwen3.8-27B: all $+0.06$ [-0.02, $+0.13$] (25:15, $p = 0.15$); clock families $+0.12$ [-0.03, $+0.22$]; sibling $+0.06$; all clock $+0.09$ [$+0.01$, $+0.17$] (16:8); control $+0.01$ (9:7); time-value $+0.06$ [$+0.01$, $+0.12$] (4:2). The pre-registered prediction was a gain on every subset; it holds on none for deepseek-v4.1-flash and only on clock points for Qwen3.8-27B.

Conventional agreement. Pointwise base scores against the execution label over the 208 separating and 80 sampled non-separating points: AUROC deepseek-v4.1-flash 0.56 (clock 0.60, control 0.46), Qwen3.5-35B-A3B 0.51 (0.56, 0.41). Qwen3.5-35B-A3B's pairwise verdicts resolved on only 35–69 of 83–125 points because of truncated outputs, and Qwen3.8-27B was paused after about 110 calls; those rows are not interpreted. The planned Best-of-N comparison over complete rollouts was not run.

R.1 TARGETED CLOCK SET

Design (fixed before the run). A constructed set, not a sample of natural tasks. Each item has a new instruction template, a new reminder content, a seeded date in 2027–2028 (every earlier group shares

one instant), one prefix step `get_current_timestamp()` whose observation is the epoch of "now", and two candidate `datetime_info_to_timestamp` calls with calendar fields; correctness comes from date arithmetic on the frozen clock, checked for every item. Type A (30 groups): both candidates fall on the target weekday, one in the coming week and one a week later; the crossed pair is "this coming Friday" against "the Friday after this coming one". Type B (30 groups): the candidates are adjacent weekdays of the coming week; the pair is "this coming Friday" against "this coming Thursday". Type C (30 items, one instruction): controls in which both candidates carry the correct date, either as integers against strings or as the correct call against an irrelevant tool. Conditions: full, render (the observation epoch annotated with date and weekday), no observation, and rerun. Verifiers: deepseek-v4.1-flash and Qwen3.5-35B-A3B under both decision-point protocols, ToolRM-Gen and ToolPRM-GRPO v4 in their pairwise formats. Predictions: (Q1) withholding the clock sends type A to chance and leaves type B near full; (Q2) render helps type A more than type B for the verifiers the crossed test flagged; (Q3) type C is unchanged. It was noted in advance that render also gives today's weekday, from which a candidate's weekday can be obtained by counting, so a smaller effect on B was predicted rather than none.

Table 15: Targeted clock set, $\mathrm{both_correct}$ on types A and B and X_A accuracy on type C; 30 anchors per type. Reruns of the two judges move every cell by at most 0.14 except Qwen3.5-35B-A3B pairwise on type B (-0.50 on $n = 18$ resolved groups).

Verifier	Protocol	A: full → render / no obs.	B: full → render / no obs.	C: full → render
deepseek-v4.1-flash	DP pointwise	1.00 → 1.00 / 0.04	1.00 → 1.00 / 0.87	0.63 → 0.60
deepseek-v4.1-flash	DP pairwise	1.00 → 1.00 / 0.20	1.00 → 1.00 / 0.97	1.00 → 1.00
Qwen3.5-35B-A3B	DP pointwise	0.64 → 0.96 / 0.00	0.56 → 1.00 / 0.12	0.73 → 0.90
Qwen3.5-35B-A3B	DP pairwise ($n = 18$ resolved)	0.78 → 1.00 / 0.31	0.78 → 0.94 / 0.72	1.00 → 1.00
ToolRM-Gen	pairwise	0.13 → 0.07 / 0.10	0.10 → 0.90 / 0.13	1.00 → 1.00
ToolPRM-GRPO v4	pairwise	0.00 → 0.07 / 0.00	0.03 → 0.87 / 0.00	0.53 → 1.00

Outcomes. Q1 holds for deepseek-v4.1-flash and fails for the other three verifiers, for which type B collapses without the clock as well: they do not know the weekday of a calendar date, so for them type B is not clock-free and the A/B contrast does not isolate the need for the observed date. Q2 fails: Qwen3.5-35B-A3B gains on both types ($+0.33$ and $+0.44$ pointwise; difference -0.10, interval $[-0.38, +0.19]$), and the trained verifiers gain on type B ($+0.80$, $+0.83$) and not on type A ($\Delta_A - \Delta_B = -0.87$ $[-1.10, -0.63]$ and -0.77 $[-0.93, -0.57]$). Q3 holds for deepseek-v4.1-flash, ToolRM-Gen, and Qwen3.5-35B-A3B pairwise and fails for ToolPRM-GRPO v4, whose full-condition failures on type C are order-inconsistent ties (14 of 30) that the annotation removes. At the cell level (post hoc), the trained verifiers under render solve the type-A row that needs only the clock (ToolRM-Gen $9/30 \to 30/30$, ToolPRM-GRPO v4 $1/30 \to 28/30$) and still pick the near date in 20–23 of 30 on the row whose phrase names the following week; type A therefore differs from type B in phrase difficulty as well as in clock need, and a matched-phrase construction is required before the week choice can be isolated.

S Additional experimental tables and figures

S.1 Inventory and observation removal

S.2 Clock representation

S.3 Scoring and aggregation

T Human audit interface

Figure 13 shows the evaluation page used for the blinded semantic audit. Raters saw the instruction, the shared history with tool observations, and two candidate next actions in randomized order. They selected which action the instruction called for, with additional options for both actions, neither action, or insufficient information, and separately judged whether the history was a reasonable start and reported confidence.

Table 16: Inventory by relation class (Table 1); every group is automatically verified. Pair: S surface-separable, F observation-free, D observation-dependent (§3.2); Pair labels come from the string-matching scorer and the observation-blind predicate (Appendix D). Families are scenario families used for clustered intervals. ToolSandbox clock families and τ^2-bench authored-prelude groups are the only groups whose T_A is not a gold replay.

Class	Family	Benchmark	Pair	Families	Groups
Stated value	stated value	BFCL/ ToolSandbox/ τ^2-bench	S	–	100 / 9 / 5
Stated value	hour (time conversion)	ToolSandbox	S	4	21
Stated value	holiday name	ToolSandbox	S	1	4
Fixed mapping	phrase → day offset	ToolSandbox	F	1	2
Observation lookup	list index	BFCL	D	6	18
Observation lookup	payment method	τ^2-bench	S	1	6
Selection by attribute	criterion over a listing	τ^2-bench/ ToolSandbox	D	5	11 / 4
Observation-based computation	date from clock, weekday edit	ToolSandbox	F	1	6
Observation-based computation	date from clock, offset edit (epoch)	ToolSandbox	D	2	3
Observation-based computation	date from clock (ISO)	τ^2-bench	D	3	5
Observation-based computation	baggage arithmetic	τ^2-bench	D	1	1
					195

Table 17: Groups solved under decision-point pairwise evaluation with the original prefix and with tool observations removed. Counts are out of n. Date and list-index judgments deteriorate, while weekday judgments remain stable. Full ablations are in Table 7.

		deepseek-v4.1-flash		Qwen3.5-35B-A3B	
Family	n	Original	No obs.	Original	No obs.
Date from clock	3	3	1	1	0
List index	7	7	0	7	0
Weekday	6	6	6	5	6
Phrase mapping	2	2	2	2	2
Hour conversion	21	21	21	21	19
Holiday name	4	4	4	4	4
Stated value	38	37	36	38	34

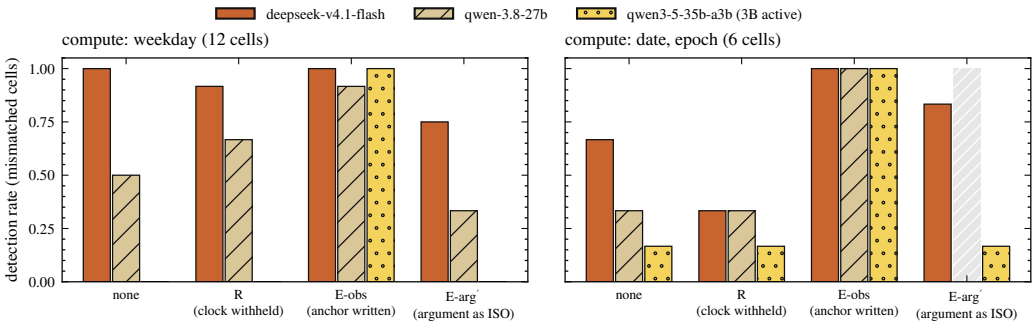

Figure 10: Evidence removal and explication on the same decisions (full-trajectory step labels; detection in mismatched cells). Mask withholds the clock observation; E-obs writes the absolute date and weekday beside the clock observation (the target date is not given); the annotation control annotates the epoch argument of a later call with an ISO date-time, leaving the observation and the judged call unchanged. Withholding the clock lowers detection only on observation-dependent pairs (dates, not weekdays); writing the clock out raises the failing judge's detection to $12/12$ and $6/6$ with matched-cell false alarms at 0 and crossed accuracy at $6/6$ and $2/3$; the annotation control changes nothing for that judge ($0/12$, $1/6$). The literal control is $8/8$ under every condition. An earlier variant that annotated the argument with its weekday exposed the answer and is not reported.

Table 18: Recovery of groups that fail at the original input, ToolSandbox clock and hour groups ($n = 36$) at the decision point. Cells show the number of failing groups that pass under each condition; *readable* changes the input on 30 of the 36 groups. Paired counts are groups recovered by one condition and not the other.

Judge	Protocol	failing	readable	caution	rerun	readable : rerun	caution : rerun
Qwen3.5-35B-A3B	pointwise	12	12	8	5	7 : 0	4 : 1
Qwen3.5-35B-A3B	pointwise, date only	2	2	0	0	–	–
Qwen3.5-35B-A3B	pairwise	3	3	1	1	–	–
Qwen3.8-27B	pointwise	7	7	6	4	3 : 0	3 : 1

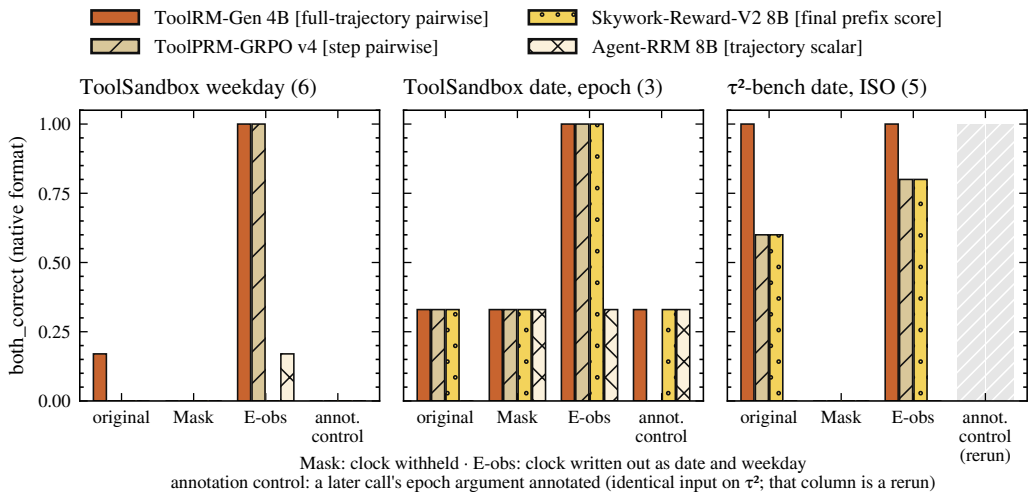

Figure 11: Clock-observation conditions for trained verifiers in their native formats (both_correct). On the tested ToolSandbox weekday and date groups, E-obs recovers the two generative verifiers; Skywork (final prefix score) recovers on dates only. The annotation control does not produce these recoveries. There are six weekday, three epoch-date, and five τ^2-bench ISO-date groups. On τ^2-bench, the annotation-control input is identical to the original input, so that column measures repeat variation. Detailed counts and output failures are in Appendix H.

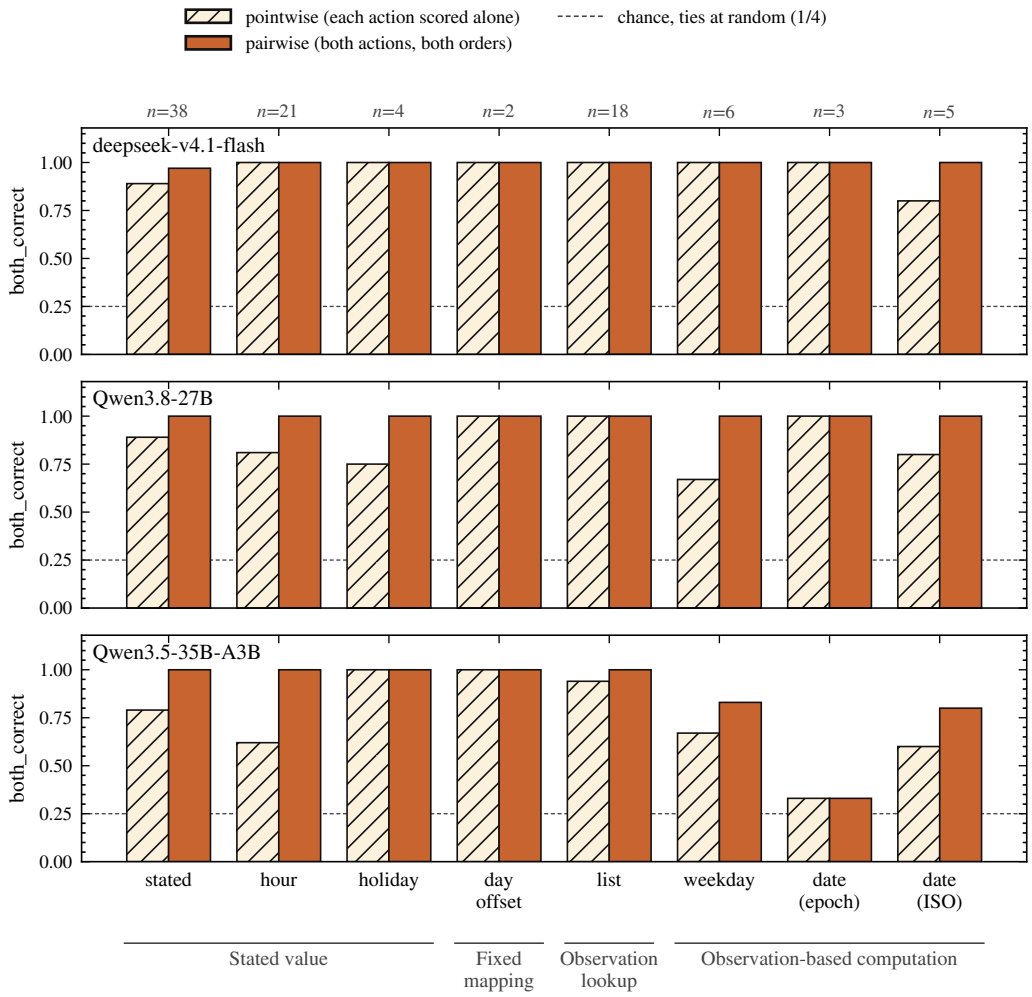

Figure 12: Decision-point crossed accuracy by relation class under pointwise scoring (hatched) and pairwise comparison (filled) of the same two candidate actions with instruction and prefix shared; relaxed scoring rule, ties broken at random, so chance is $1/4$ for both protocols (the strict two-order baseline is $1/16$). On the axis, "weekday" and "date" are the observation-free and observation-dependent pairs of observation-based computation. Pairwise is at or above pointwise in every class for both Qwen judges and equal for deepseek-v4.1-flash except on τ^2-bench dates. Group counts per class are given on the axis. Per-class McNemar on group outcomes: Qwen3.5-35B-A3B hour $b = 8$, $c = 0$, $p = 0.008$, Δ 95% CI $[+0.11, +0.58]$; stated $b = 8$, $c = 0$, $p = 0.008$, $[+0.07, +0.33]$; pooled over 92 groups $b = 19$, $c = 1$, $p < 10^{-4}$; Qwen3.8-27B pooled $b = 11$, $c = 0$, $p = 0.001$; deepseek-v4.1-flash pooled $b = 4$, $c = 1$, $p = 0.38$.

Table 19: Stated-value relations ($\mathrm{both_correct}$; tie handling is specified separately for each panel). (a) Decision-point protocols on the 38-group BFCL subset, relaxed scoring rule with ties broken at random. (b) Full-trajectory protocols on the 100 BFCL groups, with ties counted as failures and tie counts in parentheses; matched / mismatched is the step-label correctness on the first edited step, over cells with a parseable output (194–200 of 200); step labels are aggregated by minimum or mean, trajectory pairwise compares the two complete trajectories, and step pairwise compares the two divergent steps in ToolPRM-GRPO's input format. Small brackets are 95% cluster-bootstrap intervals. Skywork scores every step prefix and is read as the minimum or mean prefix score; Agent-RRM outputs one trajectory scalar (4,096-token budget); ToolRM-Gen is correct on all 114 stated-value groups. Independent noise (200 seeds) draws each step label from the judge's own label distribution, giving 0.24, 0.53, and 0.81 for Qwen3.5-35B-A3B, deepseek-v4.1-flash, and Qwen3.8-27B. The random-choice baseline is $1/4$ for pointwise and $1/16$ for two-order pairwise evaluation. ToolSandbox (11) and τ^2-bench (5) results are in Appendices H and I.

(a) Decision point (38 groups)

Verifier	Format	Pointwise	Pairwise
deepseek-v4.1-flash	judge	0.89	0.97
Qwen3.8-27B	judge	0.89	**1.00**
Qwen3.5-35B-A3B	judge	0.79	**1.00**
ToolPRM-GRPO v4	pairwise	–	0.87 [0.80, 0.93]
ToolPRM-GRPO v3	pairwise	–	0.38 [0.29, 0.48]
Sign-reversed / noise	control	0.00 / 0.24	0.00 / 0.06

(b) Full trajectory (100 groups)

Verifier	Matched	Mismatched	Labels, min (ties)	Labels, mean	Traj. pairwise	Step pairwise
deepseek-v4.1-flash	0.905	0.980	0.65 (35) [0.55, 0.74]	0.78	0.93	0.99 [0.97, 1.00]
Qwen3.8-27B	0.995	0.945	0.90 (8)	0.94	0.94	1.00
Qwen3.5-35B-A3B	0.811	0.995	0.49 (51)	0.64	0.95	0.99
ToolRM-Gen 4B	–	–	–	–	1.00	–
Skywork-RV2 8B	–	–	0.81 (18) [0.73, 0.89]	0.98	–	–
Agent-RRM 8B	–	–	0.48 (29) [0.38, 0.58]	–	–	–
Sign-reversed / noise	–	–	0.00 / 0.24–0.81	0.00 / –	0.00 / 0.06	0.00 / 0.06

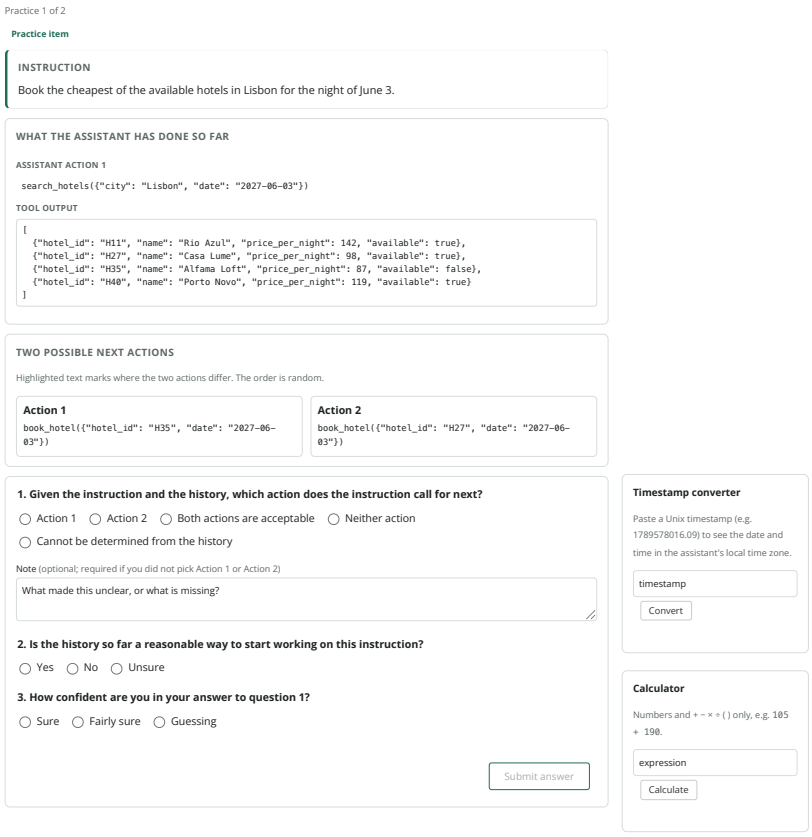

Figure 13: Interface used for the blinded semantic audit. The candidate-action order is randomized. The page asks which action the instruction calls for next, whether the preceding history is a reasonable start, and the rater's confidence; timestamp-conversion and calculator helpers are available on the side.