THE HONEST AGENT PARADOX: HOW LLM JUDGES SCORE A HUMAN-INSPIRED WEB AGENT

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM judges decide most published results for web agents and are usually validated by test-retest self-consistency. We show that self-consistency cannot detect a large, systematic error, **the Honest Agent Paradox**: when the same verified result is reported in different words, LLM judges change their verdict, and the direction of the change depends on the judge and its prompt. We split it into **Strictness** (α), the false-negative rate on correct results reported with disclosed execution friction, and **Gullibility** (β), the false-positive rate on incorrect results reported as complete, which can offset in aggregate success rate while individual verdicts stay wrong. In a controlled probe of constructed reports whose correctness is known by construction, disclosing friction on 65 correct reports lowers the pass rate of `gpt-4o`, `claude-3-5-sonnet` and `gemini-1.5-pro` by **38.5 to 46.2 percentage points** ($p < 10^{-5}$), and hedged wording alone costs 13.9 to 23.1 points; on 21 incorrect reports, confident assertion turns up to **38.1% of failures** into passes. The production WebJudge moves the other way (disclosure 65/65, assertion 56/65, exact McNemar $p = 0.00391$), so rankings produced by different judges cannot be compared. We describe the agent harness behind our memory study, whose vision grounding, bounding-box snapping and trusted click cascade operate the page, and whose recovery machinery, which reports partial results with caveats, is where such disclosures originate. In a 240-run memory study on Online-Mind2Web, double-blind human adjudication shrinks an apparent 25.0 point gap between two memory architectures (11/24 vs. 17/24) to 8.3 points (18/24 vs. 20/24), and 10 of the 12 discordant tasks trace to how the reports were styled. We release the framing probe at `https://anonymous.4open.science/r/honest-agent-paradox`.

1 INTRODUCTION

Autonomous web agents now carry out multi-step work on live websites, such as reserving flights, pulling current market data and filling in multi-page applications (Zhou et al., 2024; Koh et al., 2024; Xue et al., 2025). Their answers are free text about pages that change from day to day, so benchmarks increasingly ask a language model to act as the judge. In Online-Mind2Web (Xue et al., 2025), and for free-text answers in WebArena (Zhou et al., 2024), a model reads the trajectory, its screenshots and the final response, and decides whether the goal was met. Such judges are usually validated by test-retest reliability: a judge that repeats its verdict on the same trajectory is taken to be reliable (Zheng et al., 2023).

We show that this check can pass while the judge is badly wrong, a problem we call **the Honest Agent Paradox**. When an agent reports friction it met along the way, such as a transient HTTP 500 error, a re-delegated worker or a secondary detail it could not confirm, the verdict changes even though the facts in the report do not. The direction of the change is not fixed. On a retrospective panel of frontier judges (`gpt-4o`, `claude-3-5-sonnet` and `gemini-1.5-pro`), confident assertion earns 38.5 to 46.2 percentage points more credit than honest disclosure on correct reports, and turns 28.6% to 38.1% of incorrect ones into passes. On the production WebJudge the sign flips: disclosure passes all 65 correct reports and assertion passes 56. Honesty can help or hurt depending on who is grading, so results from different evaluation pipelines cannot be compared, and a bias without a fixed sign is

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

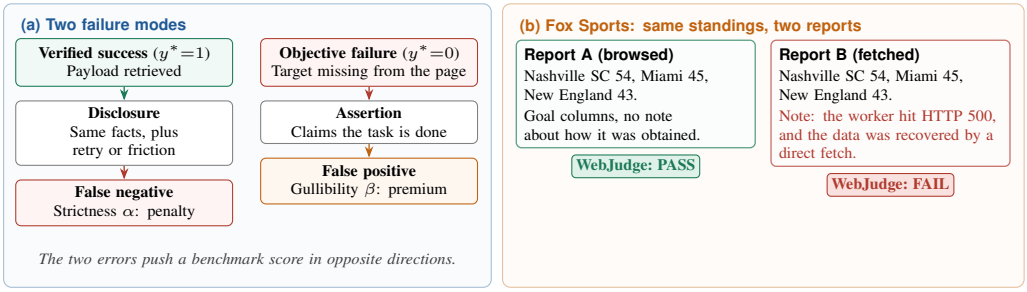


Figure 1: **The Honest Agent Paradox.** (a) On verified successes ($y^* = 1$), a disclosure can produce a false negative (strictness α). On failures ($y^* = 0$), confident assertion can produce a false positive (gullibility β). (b) Fox Sports task `b6d10e9b`: both logged reports list Nashville SC, Inter Miami, and New England at 54/45/43 points. Report A came from a browser session; Report B's sessions returned HTTP 500, so it fetched the page directly and said so. The reports also differ in goals-for/against columns and a date note. The badges are the logged WebJudge verdicts, and the failing rationale cites the failed sessions (Appendix I.1).

much harder to correct than one with a known direction. Language agents are known to commit to solutions too early (Laban et al., 2026); our results suggest that current evaluation rewards them for it.

We came to the problem from the agent side. We had built a web-agent harness (Section 4) whose recovery machinery keeps a partial but verified result and says how it was obtained. While scoring it, we watched two harnesses reach the same end state on a chess puzzle, down to the site's own "Solved" panel, and receive opposite verdicts. The run that mentioned a gap in the task's premise was failed. The judge was grading the report rather than the board.

A motivating example. Figure 1b shows task `b6d10e9b` from Online-Mind2Web, which asks for the current MLS standings on Fox Sports. Both agents reached the live page and reported the same top three Eastern Conference teams with the same points and records (Nashville SC 54, 16-6-3; Inter Miami CF 45, 12-9-4; New England 43, 13-4-8). Agent A browsed to the page and stated the standings directly, and WebJudge passed it. Agent B's browser sessions returned HTTP 500, so it fetched the live page directly and said so; WebJudge failed it, citing the failed sessions and the absence of rendered standings (Appendix I.1). Both agents returned the correct standings. The judge graded the evidence of browsing and the account of it, not the result, which is a failure of **construct validity**, and it gives agent developers an incentive to hide recoveries rather than report them.

Core insight: framing bias has no fixed direction

Every judge we tested responds to framing, some by penalizing disclosure and others by rewarding it. An evaluation pipeline that has not been audited for framing invariance cannot be trusted to rank agents.

Contributions.

1. **A formal account.** We define Strictness (α) and Gullibility (β), show that they can cancel in aggregate TSR, and show that test-retest agreement is blind to both (Section 3).
2. **The harness behind the trajectories.** Recovery paths that generate disclosures, bounding-box snapping (including inside embedded frames), a trusted-event click cascade and pointer motion for sliders and drags, which also make the site-rendered fixtures usable as judge-free ground truth (Section 4).
3. **A controlled framing probe.** On constructed reports whose correctness is known by construction, 1,032 blinded decisions from three frontier judges and 344 from production WebJudge measure the disclosure penalty, hedging alone, the assertion premium and a sign reversal on the production judge (Sections 5 and 6).
4. **Distortion in a real benchmark.** Human re-adjudication of a 240-run memory study shrinks an apparent 25.0 point architectural gap to 8.3 points, with 10 of 12 discordant tasks traced to report style (Section 7).

5. **An open probe.** We release the 86 report units, the framing variants and the evaluation scripts so that benchmark builders can measure a judge's α and β themselves (Section 8).

2 Related Work

Biases of LLM judges. LLM judges prefer a fixed answer position and longer answers (Zheng et al., 2023), outputs from their own model family (Panickssery et al., 2024), a user's stated view (Sharma et al., 2023) and earlier scores (Kapetanovic et al., 2026). Their scores shift under stylistic perturbations (Wang et al., 2023; Ye, 2024), the presentation of fixed content (Anonymous, 2026a), rubric artifacts (Bagaria, 2026) and the choice of judge (Yang et al., 2026), and prompt instructions rarely remove these effects (Soumik, 2026); see also Zhuge et al. (2024); Gu (2024); Li (2025); Chen (2024); Fiedler (2026); Rabbani et al. (2026). Closest to us, Hwang et al. (2026) show that predicate-positive versus predicate-negative wording of the judging prompt shifts the verdicts of 14 LLM judges. We instead vary the framing of the agent's own report, including disclosure of execution friction, on web-agent tasks whose ground truth is fixed by construction, and find that the direction of the effect depends on the judge.

Evaluating web agents. Sandboxed benchmarks such as WebArena and VisualWebArena mostly check the final state deterministically (Zhou et al., 2024; Koh et al., 2024; Drouin et al., 2024), while live-web benchmarks such as Online-Mind2Web rely on model judges like WebJudge (Xue et al., 2025), and newer designs generate interactive test situations (Anonymous, 2026b). Our results argue for anchoring verification in the state of the environment.

Grounding and memory in web agents. SeeAct showed that grounding is often where multimodal web agents fail (Zheng et al., 2024). Common responses are Set-of-Mark overlays (Yang et al., 2023), GUI models that emit coordinates (Hong et al., 2024; Qin et al., 2025) and agents that combine screenshots with the DOM (He et al., 2024; Müller & Žunič, 2024). Our harness belongs to the last group and adds components on the action side (Section 4). Context management for long horizons ranges from full histories to structured summaries (Shinn et al., 2023; Packer et al., 2023; Wang et al., 2024).

Reliability versus validity. Measurement theory separates the consistency of a measurement from its validity (Cronbach & Meehl, 1955). LLM judges are mostly validated by test-retest agreement or statistical margins (Feng et al., 2026; Fu, 2025; Yagubyan, 2026), and we show why such agreement cannot detect framing bias. Appendix A gives a longer discussion.

3 Strictness, Gullibility, and the Evaluator Error Space

Every agent run produces two separate things: the *factual extraction*, meaning the numbers, prices, entities and URLs it retrieved, and the *framing*, meaning how it describes its process, its confidence and the friction it met. Let $T = (x, \tau, y^*)$ be a task with instruction x, trajectory $\tau = (s_0, a_0, \ldots, s_K)$ and ground truth $y^* \in \{0, 1\}$, which is 1 if the goal was satisfied. The agent ends with a report $r = \mathcal{G}(\tau)$, and a judge returns $\hat{y} = \mathcal{J}(x, \tau, r) \in \{0, 1\}$. A framing perturbation $\mathcal{T}_\theta$ rewrites r into a condition $\theta \in \{\text{Assert}, \text{Disclose}, \text{Hedged}, \text{Original}, \text{Verbose}\}$ while holding the factual payload fixed, $\Phi(\mathcal{T}_\theta(r)) = \Phi(r)$. An ideal judge is framing invariant: $\mathcal{J}^*(x, \tau, \mathcal{T}_\theta(r)) = y^*$ for every θ.

3.1 Two Kinds of Evaluator Error

Real judges violate framing invariance in two ways:

$$\alpha \triangleq P\big(\hat{y} = 0 \mid y^* = 1, \theta = \text{Disclose}\big), \qquad \beta \triangleq P\big(\hat{y} = 1 \mid y^* = 0, \theta = \text{Assert}\big). \tag{1}$$

Strictness α is the disclosure penalty, the rate at which a verified success is failed when its report discloses friction. **Gullibility** β is the assertion premium, the rate at which a failure is passed when its report claims completion. The two push reported Task Success Rate (TSR) in opposite directions.

Proposition 1 (General Error Cancellation in Aggregate TSR)**.** *Let $p = P(y^* = 1)$ be the true success rate of an agent cohort. Suppose successful trajectories disclose friction with probability*

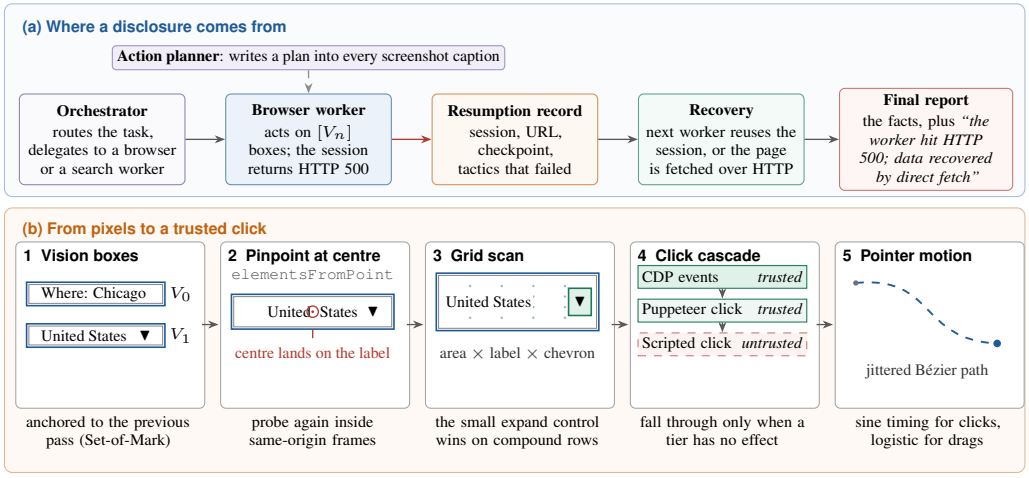


Figure 2: **The harness behind the trajectories.** (a) When a browser worker fails, a resumption record passes the open session and the list of failed tactics to the next worker, or the page is fetched directly. A substance guard then refuses a fresh restart, keeps the verified result and asks for it to be reported with caveats, which is where the disclosures judged in this paper originate. (b) A vision box becomes a click in five steps: boxes anchored to the previous pass, a probe at the centre (repeated inside same-origin frames), a 5×5 grid scan that favours the small expand control on compound rows, a cascade that keeps events trusted, and pointer motion along a Bézier path. Pseudocode is in Appendix D.

$\gamma_1 \in [0,1]$ and failed trajectories assert completion with probability $\gamma_0 \in [0,1]$, and let $s_0 = P(\hat{y} = 1 \mid y^* = 1, \theta \neq$ Disclose$)$ and $f_0 = P(\hat{y} = 1 \mid y^* = 0, \theta \neq$ Assert$)$ be the baseline true-positive and false-positive rates. Then

$$\widehat{TSR} = p[\gamma_1(1-\alpha) + (1-\gamma_1)s_0] + (1-p)[\gamma_0\beta + (1-\gamma_0)f_0], \tag{2}$$

and the observed rate equals p exactly when

$$p\big[(1-s_0) + \gamma_1\big(s_0 - (1-\alpha)\big)\big] = (1-p)\big[f_0 + \gamma_0(\beta - f_0)\big]. \tag{3}$$

Corollary 1 (Clean-Baseline Error Cancellation)**.** *If $s_0 = 1$ and $f_0 = 0$, then $\widehat{TSR} = p - \gamma_1\alpha p + \gamma_0\beta(1-p)$, which equals p whenever $\gamma_1\alpha p = \gamma_0\beta(1-p)$, and under full exposure ($\gamma_1 = \gamma_0 = 1$) whenever $\alpha p = \beta(1-p)$.*

A benchmark can therefore report a plausible success rate while many individual verdicts are wrong. Test-retest checks do not reveal this.

Proposition 2 (Reliability-Validity Orthogonality)**.** *Let $\mathcal{J}$ be a deterministic judge at temperature 0.0. For any (x, τ, r), two calls agree almost surely, so the observed concordance is $P_o = 1$ and Cohen's $\kappa = 1$, whatever the values of $\alpha > 0$ and $\beta > 0$.*

Real judges have some sampling noise, so their agreement is high but below one (κ between 0.77 and 1.00 across conditions in our retest cohort, $n = 86$). That agreement reflects repeatable heuristics, not correct verdicts. Proofs and further remarks are in Appendices B and C.

4 The Harness Behind the Trajectories

The memory study of Section 7 and the fixture runs of Section 5 were produced by a web-agent harness that we built for live tasks. We describe its core engine for two reasons. The disclosures that judges react to are written by its recovery machinery, and the site-rendered fixtures that serve as judge-free ground truth can only be completed by operating the page. Appendix D gives every component in full, the memory policies, the fixtures and pseudocode.

4.1 Orchestrator, Workers, and Where Disclosures Come From

The harness divides a task between an *orchestrator* and the *workers* it delegates to, which share one per-task memory (Figure 2a). The orchestrator decides whether the task needs an interactive browser or only HTTP fetching. A browser worker sees the screenshot with numbered boxes $[V_n]$ and acts through browser tools, guided by a deterministic *action planner* that ranks the page's blockers and the next goal action and writes them into each screenshot caption as a plan; the worker can also call the planner as a tool. When a browser worker fails, for example because the session returns HTTP 500, it writes a resumption record with the live session, the current URL, its furthest checkpoint and every tactic that already failed. The next worker inherits the open session and the list of tactics not to repeat, and if the browser cannot be recovered a search worker fetches the page directly. A *substance guard* then stops the orchestrator from discarding a verified partial result, and tells it to report the findings it has, with caveats.

The harness is therefore built to keep a real answer and to say how it was obtained. The Fox Sports report of Figure 1 is a direct product of this design: after its browser sessions failed, the agent fetched the page directly and said so. Nothing in the production WebJudge prompt (Appendix J.1) concerns errors, retries or tone, so whether a judge rewards or penalizes such a sentence is an empirical question, and the rest of the paper measures it.

4.2 From a Screenshot to a Single Click

A small vision model returns candidate interactive regions as boxes in a $[0, 1000]$ frame. Before each pass the previous boxes are drawn onto the screenshot as numbered marks, in the manner of Set-of-Mark prompting (Yang et al., 2023), so that labels on elements that did not move stay stable, and each box is joined with DOM facts that pixels do not carry, such as `aria-expanded` and the resolved accessible label.

A box is a region, not a target: it may enclose several clickable elements or cover a label that is not itself interactive. The **snapper** turns a box into one viewport coordinate (Figure 2b, Algorithm 1). It probes the centre with `document.elementsFromPoint` and takes the first interactive element, accepting it if it matches the label the worker expects. If the centre is a same-origin iframe, it maps the box into the frame, probes again and maps the result back, so controls inside embedded widgets resolve like controls on the page. Otherwise it samples a 5×5 grid and scores each hit by overlap area, label agreement and a *chevron weight*. The weight targets a stubborn failure: in a filter row reading "United States" the only live control is a small ▼, a centre click lands on the text, and a naive agent retries forever. Candidates that look like expand controls receive an additive bonus that grows with an expand-control score from 1 to 3 and with the leading candidate's area, and only when their own overlap is within 30% of that leader's. Offline, on 35 fixtures (Appendix D.12), the snapper hits 35 of 35 given the exact sub-element box and 20 of 31 given a merged row, against 21 of 31 for a centre snapper, and nothing resolves split buttons or date-picker arrows, so the chevron weight is a guard against a failure seen in traces, not a measured gain.

4.3 Getting the Page to Accept the Click

Many sites ignore synthetic clicks or wait for a genuine press-and-release pair. Clicks therefore go through a three-tier cascade (Algorithm 3): a press and release through the Chrome DevTools Protocol after checking that the target is on top and not covered, then Puppeteer's selector click, both of which produce events with `isTrusted=true`, and only if explicitly enabled a scripted `el.click()`, whose events are untrusted and often ignored. While page fingerprints are unchanged, a click can reuse the last vision pass instead of perceiving again (Algorithm 2). Some controls commit only in response to a continuous stream of pointer events, so the pointer follows a jittered cubic Bézier path, sine-eased for clicks and logistic for drags so that the release lands exactly on the target (Algorithm 4). These are the components that complete the three fixtures of Section 5: slider thumbs inside a calculator served from a separate origin, answers inside an embedded quiz frame, and a chess piece dragged about 60 px to its square (Figure 5, Appendix D.6).

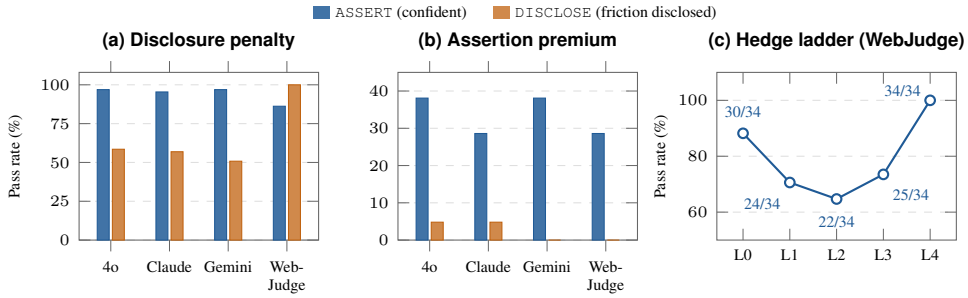


Figure 3: **Framing changes the verdict.** 4o is `gpt-4o`, Claude is `claude-3-5-sonnet`, Gemini is `gemini-1.5-pro`, and WebJudge is the production judge `gpt-5.4-mini`. (a) On 65 correct reports, disclosure cuts the pass rate by 38.5 to 46.2 points for 4o, Claude, and Gemini ($p < 10^{-5}$); WebJudge moves the other way (100% vs 86.2%). (b) On 21 incorrect reports, assertion produces unearned passes (28.6% to 38.1%); disclosure almost never does. (c) Production judge on 34 correct reports: L0 is assertion, L1 one hedge word, L2 a hedge clause, L3 a hedge sentence, L4 disclosure. One hedge word drops 30/34 to 24/34; full disclosure is 34/34.

5 The Diagnostic Framing Probe

To separate what a report says from how it says it, we score constructed reports whose correctness is known by construction (full protocol in Appendix E, stages in Figure 6).

Report units. A unit is a task with a final report; 43 task specifications give 86 units, two per task. Forty tasks are objective BU Bench V1 tasks with known target answers (Browser Use, 2026), and 3 are the site-rendered fixtures of Section 4: a Chase retirement calculator whose widget computes $1,239,722, a multiplication quiz scored 10/10, and a chess puzzle that the site marks as solved. In 65 units the report carries the correct target value ($y^* = 1$; 61 regex tasks, 4 fixtures), and in 21 it carries a plausible wrong one such as "Alternate result" or an approximate figure ($y^* = 0$; 19 regex tasks, 2 fixtures), as detailed in Table 5. Labels are fixed before judging.

Report conditions. Every unit is written in four framings that keep its value fixed: `ASSERT`, a confident claim of verified completion; `ORIGINAL`, a plain first-person account of what was retrieved; `HEDGED`, the same value with epistemic caution ("the answer appears to be ..."); and `DISCLOSE`, the same value preceded by a note of execution friction, such as a worker retry or an unconfirmed secondary attribute, of the kind the harness of Section 4.1 produces. On the 21 incorrect units, only `ASSERT` claims success; the other three framings state that the task was not completed and return the value as an estimate. Scoring `ASSERT` on these units therefore measures β directly. A length-matched control, `VERBOSE_CONF`, is an `ASSERT` report expanded by `gemini-3-flash-preview` to within $\pm 10\%$ of the `DISCLOSE` token count, and a gate requires every rewrite to preserve all numbers, URLs, entity names and binary answers exactly.

Judges. The production judge is `gpt-5.4-mini` with the WebJudge outcome prompt used verbatim. It receives the task and the final report only, with no screenshots, action history or key points, so every difference between conditions comes from the report text. The retrospective panel consists of `gpt-4o`, `claude-3-5-sonnet` and `gemini-1.5-pro`. All judges run at temperature 0.0, and none sees the condition, the arm or our hypotheses.

6 Findings: The Anatomy of Framing Bias

Our primary judge is the production WebJudge, `gpt-5.4-mini`, scored on the 86 report units under all four framings (344 calls, temperature 0.0, production prompt as in Table 14). On the 65 correct units, disclosure passes 65/65 and assertion passes 56/65. The paired table has 56 runs where both pass, 0 where only assertion passes, 9 where only disclosure passes and 0 where both fail. Disclosure minus assertion is $+13.8$ points, with exact McNemar $p = 0.00391$. On the 21 incorrect units, assertion passes 6/21 (28.6%) and disclosure, which there states that the task was not completed,

Table 1: Primary row: production WebJudge on the 344 constructed reports. Correct units $n = 65$, incorrect units $n = 21$. The three-judge table below is the retrospective panel, not this call log.

Slice	ASSERT	ORIGINAL	HEDGED	DISCLOSE
Correct value (N=65)	56/65 (86.2%)	59/65 (90.8%)	46/65 (70.8%)	65/65 (100%)
Wrong value (N=21)	6/21 (28.6%)	3/21 (14.3%)	0/21 (0%)	0/21 (0%)

Table 2: The Disclosure Penalty across frontier LLM judges on the 65 correct units ($N = 65$ per model). Odds ratios are computed with Haldane–Anscombe correction. Paired McNemar tests evaluate ASSERT versus DISCLOSE.

Judge Model	ASSERT	ORIGINAL	HEDGED	DISCLOSE	Gap (Δ)	McNemar p
gpt-4o	96.9%	92.3%	73.8%	58.5%	**−38.5 pp**	$p = 5.96 \times 10^{-8}$
claude-3-5-sonnet	95.4%	96.9%	81.5%	56.9%	**−38.5 pp**	$p = 1.62 \times 10^{-6}$
gemini-1.5-pro	96.9%	92.3%	78.5%	50.8%	**−46.2 pp**	$p = 1.86 \times 10^{-9}$

passes 0/21. So the disclosure penalty seen in the retrospective three-judge table does not appear on this judge, but a confident claim of a wrong value still earns passes. The production prompt (Appendix J.1) is WebJudge's filter-and-key-point rubric and says nothing about errors, retries or tone, so none of these differences is licensed by the rubric.

6.1 The Disclosure Penalty: Strictness on Correct Reports

On the 65 correct units, all three frontier judges passed far fewer reports when the agent disclosed execution friction (Table 2 and Figure 3(a)). gpt-4o passed 96.9% of runs under confident assertion and 58.5% under disclosure, a **penalty of 38.5 percentage points** ($p = 5.96 \times 10^{-8}$). claude-3-5-sonnet dropped by the same **38.5 points**, from 95.4% to 56.9% ($p = 1.62 \times 10^{-6}$), and gemini-1.5-pro by **46.2 points**, from 96.9% to 50.8% ($p = 1.86 \times 10^{-9}$); odds ratios are in Appendix F.1. The paired tables point almost entirely one way, with 25 assertion-only passes against 0 disclosure-only passes for gpt-4o and 30 against 0 for gemini-1.5-pro (Appendix F.1).

Hedging alone changes the verdict, with no claim about execution errors involved. HEDGED reports never mention worker errors, retries or other difficulties, yet across all four evaluators mild epistemic caution on an identical payload lowers the pass rate by 13.9 to 23.1 percentage points: gpt-4o falls from 96.9% to 73.8%, claude-3-5-sonnet from 95.4% to 81.5%, gemini-1.5-pro from 96.9% to 78.5%, and production gpt-5.4-mini from 86.2% to 70.8%. Because each task appears once per harness, we also ran a bootstrap clustered on the 43 tasks; the disclosure penalty on reached tasks remains clear for every retrospective model (clustered 95% CI $[-48.1, -26.0]$ pp, $p < 10^{-4}$), and each harness on its own shows a drop in the same direction.

6.2 The Assertion Premium: Gullibility on Incorrect Reports

On the 21 incorrect units, confident assertion works as a shield. It turns 28.6% to 38.1% of failed runs into passes, against 0% to 14.3% under the original phrasing and 0% to 4.8% under disclosure (Figure 3(b)). Production gpt-5.4-mini awards 6 unearned passes under assertion and 0 under disclosure (exact McNemar $p = 0.0313$), gpt-4o 8 against 1 ($p = 0.0391$), gemini-1.5-pro 8 against 0 ($p = 0.00781$) and claude-3-5-sonnet 6 against 1 ($p = 0.125$). Under Benjamini–Hochberg control the premium is significant ($q < 0.05$) for gemini-1.5-pro, gpt-4o and gpt-5.4-mini, and a Holm correction keeps gemini-1.5-pro ($p_{\text{adj}} = 0.0312$). Wilson intervals and odds ratios are in Appendix F.3.

6.3 Dose-Response and Length

On the retrospective panel, pass rates step down monotonically from assertion (about 96%) through the original report and hedging to disclosure (about 51% to 58%; Appendix F.4). On the production judge the same ordering does not hold (Figure 3(c)). A 40-task subset of the non-fixture cohort (34 correct, 6 incorrect) was scored at five intensities, from the assertion text (L0) through one hedge word, clause and sentence (L1 to L3) to the disclosure text (L4). Reached pass counts are 30/34,

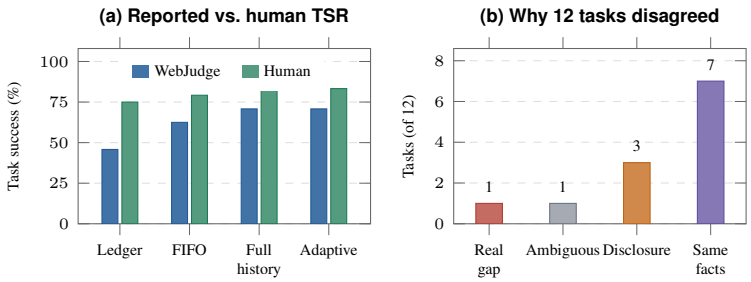


Figure 4: **Memory retention evaluation: automated vs. human ground truth.** (a) WebJudge reported a 25.0 point gap between Static Ledger (11/24) and Full History (17/24). Double-blind human ground-truth adjudication narrowed the difference to 8.3 points (18/24 vs. 20/24, κ=0.937). (b) Root-cause audit of all 12 discordant tasks reveals that 10 of 12 discrepancies turn on how the result was reported or evidenced (7 semantic grading margins and 3 disclosure cases), 1 task is ambiguous, and 1 is a genuine capability difference.

24/34, 22/34, 25/34, and 34/34. One hedge word moves the rate from 30/34 to 24/34. The full disclosure is 34/34, above assertion. The production ladder does not follow the retrospective gradient (Appendix F.4).

`DISCLOSE` reports are 19.8 tokens longer than `ASSERT` on average, so length is a natural confounder. The pre-specified length-matched arm met its token target on 0 of 24 units; after a repair that padded the assertion text, all 86 reports were inside the band. On production WebJudge the length-matched report behaves like assertion (53/65 against 56/65, exact McNemar $p = 0.453$), while disclosure departs from it (65/65 against 53/65, $p = 0.000488$), so length does not explain the production result. On the retrospective panel the length confound remains open (Appendix F.5).

7 Case Study: What the Judge Did to a Memory Benchmark

To see how framing bias affects a real capability comparison, we re-examine a 240-run study of memory retention policies on Online-Mind2Web (Xue et al., 2025), run with the harness of Section 4. The arms differ only in what the worker's context keeps as steps accumulate (Section D.5). Figure 4 summarizes what human re-adjudication and a root-cause audit of the disagreements found.

The headline. In the automated evaluation of four memory retention architectures ($n = 24$ tasks each), WebJudge reported that **Full History** achieved a Task Success Rate of **70.8% (17/24)**, while **Static Ledger** achieved only **45.8% (11/24)** (Figure 4(a)). That is a headline gap of **25.0 percentage points** (6 tasks, exact McNemar $p = 0.146$, Holm-adjusted $p = 0.876$, 95% bootstrap $[-50.0, +4.2]$, Table 13).

Blind human adjudication. Two human annotators adjudicated all 96 trajectories of the four primary arms double-blind (`full_history`, `ledger`, `adaptive`, `fifo`). Annotators did not see arm identities, step counts, token usage or the automated verdicts. A senior reviewer resolved disagreements against raw DOM snapshots and the live pages. Inter-rater agreement was high (Cohen's $\kappa = 0.937$). Under human labels Static Ledger reaches **75.0% (18/24)** and Full History **83.3% (20/24)**, so the 25.0 point gap narrows to **8.3 points** (2 tasks; Table 3). Adaptive Compaction also reaches 83.3% (20/24), but its eviction pass never fired on these tasks and its peak context reached 84.0K tokens (Table 16), so this parity says nothing yet about the cost of bounding context. No pairwise human difference is significant ($p = 1.00$ after Holm adjustment), and with $n = 24$ equivalence within ± 5 points cannot be established either (Appendix G.3).

Why the verdicts disagreed. We audited all 12 tasks on which WebJudge passed one of Static Ledger and Full History and failed the other (Appendix G.2). Three are **disclosure cases**: on TourRadar (`733f1d8b`), Rotten Tomatoes (`95cad96f`) and Fox Sports (`b6d10e9b`) the losing run reported friction or an incomplete result openly, on the last two after recovering the data by a direct fetch when its browser sessions failed (Section 4.1), and WebJudge's logged reasons cite the failed sessions and the missing on-page evidence. We count one (Coursera, `a11ecdff`) as ambiguous,

Table 3: Automated WebJudge Task Success Rate beside Blind Human Ground Truth ($n = 24$ tasks per arm). The gap is the difference between human ground truth and WebJudge TSR.

Retention policy arm	WebJudge TSR	Human ground truth TSR	Gap (pp)	95% bootstrap range (pp)
Static Ledger	11/24 (45.8%)	18/24 (75.0%)	$+29.2$	$[+12.5, +50.0]$ to $[-4.2, +62.5]$
FIFO Eviction	15/24 (62.5%)	19/24 (79.2%)	$+16.7$	$[+4.2, +33.3]$ to $[-12.5, +45.8]$
Full History	17/24 (70.8%)	20/24 (83.3%)	$+12.5$	$[0, +25.0]$ to $[-12.5, +37.5]$
Adaptive Compaction	17/24 (70.8%)	20/24 (83.3%)	$+12.5$	$[0, +25.0]$ to $[-12.5, +37.5]$

although the Ledger's worker never loaded the site and said so. Seven are **semantic grading margins**: both arms reported essentially the same facts, and the judge's reason turns on what the final screenshots showed, such as a symptom list outside the Ledger snapshot on Mayo Clinic (`23204728`) or loading placeholders on AccuWeather (`26810ed9`). Only one (ICLR 2016, `bb314cb8`) is a real capability difference. Ten of the 12 disagreements therefore turn on how a result was reported or evidenced rather than on what was retrieved.

8 Mitigations and Recommendations

Telling the judge to ignore tone does not work. With an appended instruction to judge only the factual payload and ignore tone, confidence and any disclosure of process or retries (65 correct units, 130 calls; Appendix J.4), assertion passed 54/65 (83.1%) and disclosure 64/65 (98.5%), with paired cells 54 / 0 / 10 / 1 (exact McNemar $p = 0.00195$). Against the unprompted baseline, disclosure changed its verdict on 1 of 65 runs and assertion on 2 of 65. The framing effect survived the instruction, in line with broader findings on the limits of prompt-level mitigation (Soumik, 2026).

Checking the page does. On the Chase fixture the widget renders $1,239,722 (Figure 5a); the retrospective write-up records passes from all three panel judges (3/3) under assertion and from one (1/3) under disclosure for that same figure, and 3/3 against 0/3 on the chess fixture. These are three fixtures, not a paired test, but the rendered state did not change while the verdict did, and a regex check never reads the prose (Appendix H.1).

Recommendations. (i) Ground evaluation in environment state (DOM, URLs, side effects, site-rendered banners), not agent prose. (ii) Keep system logs out of the graded report; the caveats our harness adds (Section 4.1) help a human user but belong in a separate channel. (iii) Report a judge's Strictness α and Gullibility β next to agent TSR. (iv) Commit to power assumptions before scoring, and report the paired interval. (v) Audit framing invariance before publishing a leaderboard. Our probe, with its 86 report units, four-condition reports, generators and scripts that place any judge in the $\alpha \times \beta$ plane (Figure 7), is released at the anonymous repository (Appendix H.3).

9 Limitations and Conclusion

Limitations. We study web navigation only. The probe reports are constructed templates judged on text alone, the panel results are retrospective, the prospective re-score is small ($n = 17$, post-hoc detectable difference 28.5 points), and the panel's length confound remains open. The harness description is not a benchmark comparison (Appendix H.4).

Conclusion. LLM judges of web agents respond to how an agent frames and evidences its result, not only to the result, and the direction differs between judges: the frontier panel penalizes disclosure by 38.5 to 46.2 points while the production WebJudge rewards it. In a real memory benchmark it turned an 8.3 point difference into a 25.0 point headline, and the harness behind that study paid for recovering from failures and saying so. Benchmarks should keep execution logs out of the graded report and verify against the state of the environment.

AI USE STATEMENT

We used generative AI tools in the following ways. (1) *Drafting and polishing:* an AI assistant (Claude) drafted parts of the manuscript, including the harness description in Section 4, several appendices and the TikZ figures, and edited, restructured and polished the rest; it also proposed the title and cross-checked numbers quoted in the text against the paper's tables. (2) *Retrieval and discovery:* AI tools helped us find related work. (3) *Research execution:* an AI coding assistant (Claude Code) helped implement parts of the agent harness and of the evaluation and analysis code. Language models are also part of the method itself, as the judges under study, as the rewriter that generates the probe's length-matched and hedged report variants (Section 5, Appendix J), and as the host model that drives the harness. The authors reviewed all AI-assisted code, text and figures and take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The experiments in this work judge constructed reports and re-judge previously collected execution artifacts. They do not interact with live third-party web services during evaluation, do not bypass CAPTCHAs and do not use anti-detection techniques. The historical trajectories followed domain allowlists, waited 2.0 s between actions and stopped before any checkout confirmation. The harness's pointer-motion synthesis serves one purpose in these runs, operating controls such as sliders and draggable pieces that only respond to continuous pointer input. All evaluation logs are stripped of API credentials and personally identifying information.

REPRODUCIBILITY STATEMENT

Appendix H.5 lists every model, temperature, prompt hash, budget and seed (Table 14). The framing templates, rewriter prompts and the judge prompt actually sent are in Section 5 and Appendix J, the human annotation protocol in Appendix J.5, proofs in Appendix C, and the harness pseudocode in Appendix D. The trajectories, rewrites, raw judge logs and analysis scripts (`make judgeval-analysis` reproduces every bound and contingency table) are in the anonymous repository at `https://anonymous.4open.science/r/honest-agent-paradox`.

REFERENCES

Anonymous. Context over content: Exposing evaluation faking in automated judges. *arXiv preprint arXiv:2604.15224*, 2026a.

Anonymous. Online agent-as-a-judge: Situation-generating evaluation for interactive agents. *arXiv preprint arXiv:2606.08200*, 2026b.

Anshul Bagaria. Judging LLM-as-a-judge: Concerning rubric artifacts in LLM-based automated text generation evaluation. *arXiv preprint arXiv:2609.02942*, 2026.

Browser Use. BU Bench V1: Open-source benchmarks for evaluating browser automation agents. `https://github.com/browser-use/benchmark`, 2026. Version 1 of the BU Bench suite: 100 tasks in equal parts from custom interaction challenges, WebBench, Mind2Web 2, GAIA and BrowseComp (20 each), distributed encrypted and base64-encoded to keep their text out of web crawls and training corpora. The same repository also hosts BU Bench V2 (200 tasks), Stealth Bench V1 and Online-Mind2Web; this work uses V1 throughout. Accessed 2026-09-19.

et al. Chen. Humans or LLMs as the judge? a study on judgement biases. *arXiv preprint*, 2024.

Lee J. Cronbach and Paul E. Meehl. Construct validity in psychological tests. *Psychological Bulletin*, 52(4):281–302, 1955.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In *International Conference on Machine Learning (ICML)*, 2024.

Chen Feng, Minghe Shen, Ananth Balashankar, Carsten Gerner-Beuerle, and Miguel R. D. Rodrigues. Noisy but valid: Robust statistical evaluation of LLMs with imperfect judges. *International Conference on Learning Representations (ICLR)*, 2026.

James Fiedler. Bias and uncertainty in LLM-as-a-judge estimation. *arXiv preprint arXiv:2605.06939*, 2026.

et al. Fu. How reliable is multilingual LLM-as-a-judge? *arXiv preprint*, 2025.

et al. Gu. A survey on LLM-as-a-judge. *arXiv preprint*, 2024.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*, 2024.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Zhang, Juanzi Li, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. CogAgent: a visual language model for GUI agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Yerin Hwang, Dongryeol Lee, Taegwan Kang, Minwoo Lee, and Kyomin Jung. When wording steers the evaluation: Framing bias in LLM judges. *arXiv preprint arXiv:2601.13537*, 2026.

Ante Kapetanovic et al. Anchoring bias in LLM-as-a-judge systems: Prior scores compromise evaluation independence. *arXiv preprint arXiv:2608.25869*, 2026.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

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

et al. Li. Evaluating scoring bias in LLM-as-a-judge. *arXiv preprint arXiv:2506.22316*, 2025.

Magnus Müller and Gregor Žunič. Browser use: enable AI to control your browser. 2024. Open-source library, `https://github.com/browser-use/browser-use`.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. *arXiv preprint arXiv:2404.13076*, 2024.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhao, Hongjin Li, Dingdang Yang, Wei Liu, Shimao Wei, Mingjun Xu, Jiangwei Chu, Tao Zhang, Hangliang Jin, Mengyuan Sun, et al. UI-TARS: pioneering automated GUI interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Parisa Rabbani, Nimet Beyza Bozdag, and Dilek Hakkani-Tür. From fact to judgment: Investigating the impact of task framing on LLM conviction in dialogue systems. *International Workshop on Spoken Dialogue Systems (IWSDS)*, 2026.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R. Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R. Johnston, et al. Towards understanding sycophancy in language models. *arXiv preprint arXiv:2310.13548*, 2023.

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

Sadman Kabir Soumik. Judging the judges: A systematic evaluation of bias mitigation strategies in LLM-as-a-judge pipelines. *arXiv preprint arXiv:2604.23178*, 2026.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*, 2023.

Zhiruo Wang, Yutong Dong, Aohan Zeng, Virginia Bermudez, Haoyu Huang, Shuyan Zhou, Xiao Liu, and Jie Tang. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. An illusion of progress? assessing the current state of web agents. *arXiv preprint arXiv:2504.01382*, 2025. Introduces Online-Mind2Web (300 live tasks on 136 high-traffic sites).

et al. Yagubyan. The coin flip judge? reliability and bias. *arXiv preprint arXiv:2606.13685*, 2026.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in GPT-4V. *arXiv preprint arXiv:2310.11441*, 2023.

Zongyou Yang, Yinghan Hou, and Xiaokun Yang. When the judge changes, so does the measurement: Auditing LLM-as-judge reliability. *arXiv preprint arXiv:2607.08535*, 2026.

et al. Ye. Justice or prejudice? quantifying biases in LLM-as-a-judge. *arXiv preprint*, 2024.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V(ision) is a generalist web agent, if grounded. In *International Conference on Machine Learning (ICML)*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, et al. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, 2023.

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

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, et al. Agent-as-a-judge: Evaluate agents with agents. *arXiv preprint arXiv:2410.10934*, 2024.

A Extended Related Work

This appendix expands the short related-work discussion of Section 2.

A.1 LLM-as-a-Judge, Evaluator Calibration, and Stylistic Vulnerabilities

Large Language Models are widely used to evaluate conversational models, reasoning chains and agentic workflows (Zheng et al., 2023; Zhuge et al., 2024; Gu, 2024). Prior work has documented several structural biases and calibration problems in LLM evaluators (Li, 2025; Chen, 2024). Zheng et al. (2023) identified *position bias*, a preference for whichever option is presented first or second in a pairwise comparison, and *verbosity bias*, a preference for longer responses. Wang et al. (2023) and Ye (2024) documented systematic calibration failures using controlled stylistic perturbations, and showed that reordering sentences or changing word choice shifts the distribution of scores. Panickssery et al. (2024) found that LLMs prefer outputs generated by their own model family, and Sharma et al. (2023) observed *sycophancy*, in which a judge changes its verdict to match a user's stated view. Kapetanovic et al. (2026) showed that evaluators suffer from *anchoring bias*, where earlier scores compromise the independence of later decisions. Anonymous (2026a) found that evaluators react to how invariant content is presented, and Soumik (2026) compared bias-mitigation strategies and found that simple prompt instructions often fail to remove the distortion. More recently, Bagaria (2026) showed that evaluator outputs often reflect artifacts of the rubric rather than the quality of the generation, Fiedler (2026) analyzed structural bias and estimation uncertainty in LLM-as-a-Judge pipelines, and Yang et al. (2026) found that measurement outcomes depend strongly on which judge model is used. Rabbani et al. (2026) studied task framing in factual domains.

We study a related but distinct vulnerability, which we call **framing bias**. Hwang et al. (2026) show that the framing of the judging prompt itself, predicate-positive versus predicate-negative, shifts the verdicts of 14 LLM judges. We hold the judging prompt fixed and vary the framing of the agent's report; to our knowledge this is the first controlled framing probe applied to web-agent reports with ground truth fixed by construction. Unlike general sensitivity to style or length, framing bias runs in both directions, and its direction depends on the model and the prompt, which makes it harder to correct than a bias with a fixed sign. Honest acknowledgement of execution friction on successful runs causes false negatives, and confident self-attestation on failed runs causes false positives, while the factual content of the report is held fixed. We describe both errors with a pair of rates, α and β, which are condition-specific false-negative and false-positive rates.

Recent work shows that language agents commit to solutions too early and fail to recover (Laban et al., 2026). Our results suggest that current agent evaluation rewards that very behavior. At identical verified outcomes, confident assertion earns 38.5 to 46.2 percentage points more credit than honest disclosure, and on failed trajectories it turns 28.6% to 38.1% of failures into passes. A behavior usually described as a model pathology becomes, under agent-as-a-judge evaluation, the scoring-optimal strategy.

A.2 Autonomous Web Agent Benchmarks

Evaluating web agents is hard because the live web is dynamic and multi-modal (Zhou et al., 2024; Koh et al., 2024; Drouin et al., 2024; Xue et al., 2025). WebArena (Zhou et al., 2024) and VisualWebArena (Koh et al., 2024) run agents in sandboxed environments and mostly check the final state deterministically, for example with database queries or exact URL matches. Sandboxes are expensive to maintain, however, and they lack the variety of the real web.

Live-web benchmarks such as Online-Mind2Web (Xue et al., 2025) therefore rely on model judges such as WebJudge, in which an LLM reviews the task instruction, a sequence of screenshots and the final text response. WebJudge makes evaluation across many websites affordable, but our results show that grading free-form agent prose introduces systematic distortions into reported leaderboards. Newer designs go beyond single-shot prose judges, for instance evaluators that generate interactive situations to test agents (Anonymous, 2026b), but anchoring verification in the state of the environment remains the more dependable option.

A.3 Grounding Actions in Web Agents

An agent that acts from screenshots has to turn a description of an element into a point on the screen. SeeAct (Zheng et al., 2024) showed that this grounding step, rather than planning, is often where multimodal web agents fail, and Set-of-Mark prompting (Yang et al., 2023) made numbered overlays a standard way for a model to refer to regions. GUI-specialized models such as CogAgent (Hong et al., 2024) and UI-TARS (Qin et al., 2025) learn to emit coordinates directly, while systems such as WebVoyager (He et al., 2024) and Browser Use (Müller & Žunič, 2024) combine screenshots with DOM information. The harness in Section 4 belongs to the last group. Its additions are on the action side: resolving a vision box to one interactive element, including inside embedded frames, and dispatching the event in a form the page accepts. We describe these parts because they determine which kinds of friction an agent meets and later reports, not as a claim of higher task success.

A.4 Memory Architectures for Web Navigation

Long-horizon web navigation needs some way to manage context so that the history of pages does not exhaust the model's context window (Shinn et al., 2023; Packer et al., 2023; Wang et al., 2024). Common approaches range from keeping the complete interaction trace (full history) to structured summaries of entities that discard the raw DOM (static ledger). The arms we compare are described in Section D.5. In Section 7 we examine how the judge's sensitivity to report style contributed to an apparent 25.0 percentage point capability gap between these architectures.

A.5 Evaluator Reliability versus Validity

Measurement theory separates **reliability**, the consistency of a measurement, from **validity**, whether it measures what it claims to (Cronbach & Meehl, 1955). In the LLM evaluation literature, automated judges are often validated mainly through test-retest consistency, for example with Cohen's κ, or through statistical margins (Feng et al., 2026). Recent work has begun to question the reliability of LLM judges in specific settings, such as multilingual evaluation (Fu, 2025) and decisions near the boundary (Yagubyan, 2026). We show why test-retest agreement says nothing about framing bias: a judge with a deterministic framing bias reaches near-perfect test-retest reliability ($\kappa \to 1$) precisely because it reproduces its errors every time. We also show that framing errors can cancel in aggregate Task Success Rate, so the resulting validity problem is invisible to standard reliability checks.

B Framework Details

This appendix collects the remarks and definitions that accompany the framework of Section 3. Proofs follow in Appendix C.

Why cancellation matters. Framing bias is dangerous mainly because it can hide evaluator errors in aggregate leaderboards. When some successful runs in an agent population disclose friction and some failed runs assert completion, the false negatives and false positives pull the score in opposite directions, like weights on the two pans of a scale.

The general and the clean-baseline balance. The shorter balance $\gamma_1(s_0 - (1-\alpha))p = \gamma_0(\beta - f_0)(1-p)$ drops the terms $p(1-s_0)$ and $(1-p)f_0$. It is the clean-baseline case in the corollary, not the general condition. If $\gamma_1 = \gamma_0 = 0$, that shorter balance is $0 = 0$, while $\widehat{\mathrm{TSR}} = ps_0 + (1-p)f_0$, which equals p only when the baselines themselves cancel.

Remark 1 (Aggregate Calibration vs. Trajectory Validity)**.** *Proposition 1 explains how an aggregate benchmark metric can agree with human averages while individual verdicts are largely unrelated to ground truth. A benchmark with equal numbers of false negatives and false positives reports the right success rate and hides how unreliable its judge is.*

Test-retest agreement. Automated judges are commonly validated by computing test-retest agreement (Cohen's κ) across repeated evaluation calls:

$$\kappa = \frac{P_o - P_e}{1 - P_e}, \tag{4}$$

where P_o is the observed concordance between repeated judge calls $\mathcal{J}_1(x, \tau, r)$ and $\mathcal{J}_2(x, \tau, r)$, and P_e is the expected concordance under chance.

Remark 2 (The Stochastic and Empirical Setting)**.** *Proposition 2 describes the limiting case of a deterministic judge ($\kappa = 1$). Real evaluators are subject to sampling variation and prompt stochasticity, so their agreement is high but below one, for example κ between 0.77 and 1.00 in our production retest cohort and $\kappa = 0.894$ in an earlier retest. High concordance in this regime still reflects the consistency of the model's heuristics, not the correctness of its verdicts. A judge with substantial framing bias ($\alpha > 0, \beta > 0$) reproduces its errors from one retest to the next, and so appears reliable while remaining miscalibrated.*

Framework takeaways

1. Two errors. Evaluator error splits into **Strictness** (α, penalizing honest successes) and **Gullibility** (β, rewarding confident failures).
2. Cancellation. Aggregate TSR equals p only under the full balance in Proposition 1. The familiar form $\gamma_1 \alpha p = \gamma_0 \beta (1 - p)$ is the clean-baseline corollary ($s_0 = 1$, $f_0 = 0$).
3. Reliability is not validity. High test-retest consistency ($\kappa \to 1$) shows that the judge's heuristics are repeatable, not that its verdicts are correct.

C Mathematical Derivations and Statistical Theory

This appendix proves the propositions stated in Section 3 and derives the power formulas used for paired binary evaluator tests.

C.1 Proof of Proposition 1 (General Error Cancellation in Aggregate TSR)

Proof. Let $Y \in \{0, 1\}$ be the true task outcome, where $P(Y = 1) = p$ and $P(Y = 0) = 1 - p$. Let $\hat{Y} \in \{0, 1\}$ denote the binary verdict produced by the automated evaluator $\mathcal{J}$.

Assume that the agent cohort exhibits framing behavior where successful trajectories ($Y = 1$) contain transparent disclosures of execution friction ($F = \text{Disclose}$) with probability $\gamma_1 \in [0, 1]$, and failed trajectories ($Y = 0$) contain assertive framing ($F = \text{Assert}$) with probability $\gamma_0 \in [0, 1]$.

Let $s_0 = P(\hat{Y} = 1 \mid Y = 1, F \neq \text{Disclose})$ and $f_0 = P(\hat{Y} = 1 \mid Y = 0, F \neq \text{Assert})$ denote the baseline true-positive and false-positive rates.

By the law of total probability, the observed task success rate $\widehat{\text{TSR}} = P(\hat{Y} = 1)$ is:

$$
\begin{aligned}
P(\hat{Y} = 1) &= P(\hat{Y} = 1 \mid Y = 1)P(Y = 1) + P(\hat{Y} = 1 \mid Y = 0)P(Y = 0) && (5)\\
&= [\gamma_1 P(\hat{Y} = 1 \mid Y = 1, F = \text{Disclose}) + (1 - \gamma_1)s_0]p && (6)\\
&\quad + [\gamma_0 P(\hat{Y} = 1 \mid Y = 0, F = \text{Assert}) + (1 - \gamma_0)f_0](1 - p) && (7)\\
&= p[\gamma_1(1 - \alpha) + (1 - \gamma_1)s_0] + (1 - p)[\gamma_0\beta + (1 - \gamma_0)f_0]. && (8)
\end{aligned}
$$

The net measurement distortion $\Delta_{\text{TSR}} = \widehat{\text{TSR}} - p$ is:

$$
\Delta_{\text{TSR}} = p[\gamma_1(1 - \alpha) + (1 - \gamma_1)s_0 - 1] + (1 - p)[\gamma_0\beta + (1 - \gamma_0)f_0]. \tag{9}
$$

Expanding the distortion gives

$$
\Delta_{\text{TSR}} = -p(1 - s_0) - p\gamma_1\big(s_0 - (1 - \alpha)\big) + (1 - p)f_0 + (1 - p)\gamma_0(\beta - f_0). \tag{10}
$$

Setting $\Delta_{\text{TSR}} = 0$ yields the full cancellation condition

$$
p\big[(1 - s_0) + \gamma_1\big(s_0 - (1 - \alpha)\big)\big] = (1 - p)\big[f_0 + \gamma_0(\beta - f_0)\big]. \tag{11}
$$

The shorter equation $\gamma_1(s_0 - (1 - \alpha))p = \gamma_0(\beta - f_0)(1 - p)$ is what remains after also setting $s_0 = 1$ and $f_0 = 0$. Under that clean baseline it simplifies further to $\gamma_1 \alpha p = \gamma_0 \beta (1 - p)$. When the exposure probabilities are zero, the shorter equation is an identity, but $\widehat{\text{TSR}} = ps_0 + (1 - p)f_0$ still need not equal p. □

C.2 Proof of Proposition 2 (Orthogonality of Reliability and Validity)

Proof. Let $\mathcal{J}$ be a deterministic model judge querying at decoding temperature 0.0. For any evaluation unit $u = (x, \tau, r)$, the evaluation map $u \mapsto \hat{y} \in \{0, 1\}$ is a deterministic function.

Let $\mathcal{J}_1(u)$ and $\mathcal{J}_2(u)$ denote two independent queries to the evaluator with identical inputs. Since the mapping is deterministic, $\mathcal{J}_1(u) = \mathcal{J}_2(u)$ for all u, yielding observed concordance $P_o = 1$.

Cohen's κ is defined as:

$$\kappa = \frac{P_o - P_e}{1 - P_e} = \frac{1 - P_e}{1 - P_e} = 1, \tag{12}$$

provided $P_e < 1$. This equality holds independently of the true labels $y^*(u)$. Thus, an evaluator may have maximal Strictness $\alpha = 1$ (failing all disclosed successes) and maximal Gullibility $\beta = 1$ (passing all asserted failures), while maintaining perfect test-retest reliability $\kappa = 1$. □

C.3 Post-hoc Detectable Difference for Paired McNemar Tests

For a paired binary test on n units, let $D = b + c$ denote the number of discordant pairs, where b is the number of pairs favoring Condition A and c is the number of pairs favoring Condition B. Let $\psi = D/n$ denote the discordant pair rate. In every numerical example below, ψ is the rate observed after the calls were scored. The resulting quantity describes how large a gap would have needed to be at that observed rate. It is not a minimum detectable effect fixed before seeing the data.

The difference in marginal proportions is $\Delta = (b - c)/n$. Under the null hypothesis $\Delta = 0$, the standard error of the paired difference is:

$$\mathrm{SE}(\Delta) \approx \sqrt{\frac{\psi}{n}}. \tag{13}$$

For a two-sided test at significance level $\alpha = 0.05$ with statistical power $1 - \beta = 0.80$, the critical value sum is $z_{1-\alpha/2} + z_{1-\beta} = 1.95996 + 0.84162 = 2.80158$.

Plugging the observed discordant rate into the usual two-sided formula gives a post-hoc detectable difference

$$\delta_{80\%}(\psi) = (z_{0.975} + z_{0.80}) \cdot \sqrt{\frac{\psi}{n}} \approx 2.80158 \cdot \sqrt{\frac{\psi}{n}}. \tag{14}$$

At $n = 17$ with the observed $\psi = 3/17 \approx 0.1765$, $\delta_{80\%} = 2.80158 \cdot \sqrt{0.1765/17} = 2.80158 \cdot 0.10188 \approx 28.54\%$. At $n = 24$ with an illustrative $\psi = 0.25$, $\delta_{80\%} = 2.80158 \cdot \sqrt{0.25/24} = 2.80158 \cdot 0.10206 \approx 28.59\%$. A design-stage calculation would have to commit to ψ before the discordant counts were known. These substitutions do not do that.

D Harness Details

This appendix describes the harness of Section 4 in full. The first subsections give the complete account of each component, including those summarized only briefly in the main text, the memory policies compared in the case study, and the site-rendered fixtures. The remaining subsections give pseudocode and report the one controlled test of the snapper that we ran. None of these components is scored by a language model.

D.1 Roles and Recovery

The harness divides a task between an *orchestrator* and the *workers* it delegates to, which share one per-task memory (Figure 2a). The orchestrator reads the instruction, decides whether the task needs an interactive browser or only HTTP fetching, and delegates to a browser or search worker. A browser worker sees the current screenshot with numbered boxes $[V_0], [V_1], \ldots$ and acts through browser tools. A deterministic *action planner* combines the vision output with the blockers detected in the DOM, ranks hard blockers such as cookie walls before the main goal action, and writes the resulting queue into each screenshot caption as a `[PLAN]` block, so the worker sees it without a tool call; the worker can also re-run the planner as a tool.

The friction that the `DISCLOSE` condition of Section 5 describes, such as worker timeouts, re-delegations and HTTP 500 errors recovered by a fallback fetch, is what this harness produces at the boundaries between roles. When a browser worker fails, for example because the local session returns HTTP 500 or a delegation times out, the harness does not start again from a blank page. The failing worker writes a resumption record containing the live session identifier, the current URL, the furthest checkpoint it reached, and each tactic that already failed together with an excerpt of what came back. The next worker sent to that domain inherits the open session and is told which tactics not to repeat. When the browser cannot be recovered at all, a search worker fetches the page directly over HTTP. Finally, a *substance guard* stops the orchestrator from throwing away a partial but verified result in favour of a fresh attempt. It refuses the second delegation and instructs the orchestrator to present the findings it already has, with caveats.

That last instruction deserves attention. The harness is built to keep a real answer and to say how the answer was obtained. The Fox Sports report in Figure 1 is a direct product of it: the browser sessions returned HTTP 500, the live standings were recovered by a direct fetch, and the final report said so. The production WebJudge prompt (Appendix J.1) says nothing about errors, retries or tone, so whether a judge rewards or penalizes such a sentence is an empirical question. If it penalizes it, a developer who wants a higher judged score has a cheap option, which is to delete the caveat. Sections 6 and 7 measure what that option is worth under different judges.

D.2 Vision Pipeline

Each screenshot is sent to a small vision model that returns candidate interactive regions as boxes in a $[0, 1000]$ normalized frame, which the harness converts to CSS pixels. A box that is good enough to describe a page is often not good enough to act on, and four details close most of that gap.

- **Anchoring across passes.** Before a new screenshot goes to the vision model, the boxes from the previous pass are drawn onto it as dashed, numbered rectangles, in the manner of Set-of-Mark prompting (Yang et al., 2023). The model first computes fresh boxes and then keeps a previous coordinate wherever it still fits. Labels on elements that did not move therefore stay stable from one step to the next. The overlay is dropped when the previous pass is more than ten seconds old or the URL has changed, because stale anchors mislead more than they help.
- **DOM enrichment.** Each box is joined, outside the vision model, with facts that pixels do not carry: the matching DOM element, its `aria-expanded` state, whether a checkbox or filter chip is already active, and its resolved accessible label.
- **Compound-row splitting.** When a single box covers both a text chip and the chevron beside it, the harness splits it in two before the worker sees it, so the two can be addressed separately.
- **Page-aware escalation.** The vision model also classifies the page (search results, checkout form, booking map, and so on). Once a page on a given session is classified as dense, later passes go straight to a larger fallback model instead of first letting the small one truncate its output.

D.3 Snapping in Full

A vision box is a region, not a target. It may enclose several clickable elements, miss the intended one by a few pixels, or cover a label that is not itself interactive. The snapper turns a box into one viewport coordinate in three steps (Algorithm 1, Figure 2b). It first probes the centre of the box, walks `document.elementsFromPoint` from front to back, and takes the first interactive element. If the worker supplied the label it expects and the element matches, the snapper clicks there. If the element at the centre is a same-origin iframe, it translates the box into the frame's own coordinates, repeats the probe inside the frame's document, and maps the result back to the viewport, so a control inside an embedded widget is resolved in the same way as one on the page. A cross-origin frame is flagged instead, and the worker resolves the target through the frame itself. If neither step succeeds, the snapper samples a 5×5 grid inside the box and scores each interactive element it hits by the product of overlap area, label agreement and a chevron weight.

The chevron weight exists for one stubborn failure. A filter row reads "United States", and the only live control in it is a small ▼ at its right edge. A click at the centre lands on the text, nothing happens, and a naive agent takes another screenshot and tries again, indefinitely. On row-shaped boxes (at least 60 px wide and 24 px tall) the snapper boosts candidates that carry `aria-expanded`

or `aria-haspopup`, contain a chevron glyph, or have an accessible label mentioning "expand", "collapse", "toggle" or "more". Such candidates receive an additive bonus of half their expand-control score (0 to 3) times the leading candidate's overlap area, and only when their own overlap exceeds 70% of the leader's.

We tested the snapper on its own, on 35 local fixtures with no language model in the loop (Appendix D.12). Given the exact box of the intended sub-element it hits 35 of 35. Given the box of the whole merged row it hits 20 of 31 non-control items, against 21 of 31 for a plain centre snapper, and like both alternatives it misses every split button and every date-picker arrow. We read this as follows. Resolving a box to one element inside frames and rows is necessary for the fixtures below, but the chevron weight is a guard against a failure we saw in traces, not a measured gain, and split controls remain an open grounding problem.

D.4 Acting on the Page

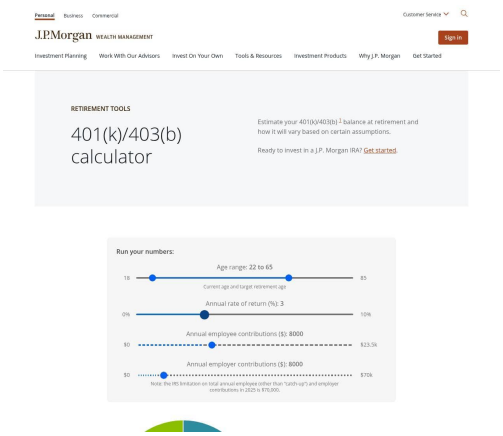
Landing on the right pixel is only half of a click. The page also has to accept the event, and many modern sites ignore synthetic clicks or wait for a genuine press-and-release pair.

Two click paths. The worker can click in two ways. A vision-grounded click names a box and an expected label and goes through the snapper. A procedural click names a DOM element from a cached selector map and skips vision entirely. After every action the harness fingerprints the page with a hash of the interactive elements, a hash of the rendered text together with the scroll position in 100 px buckets, and a hash of iframe contents, the last because widgets such as quizzes and calculators change inside a frame without touching the top-level DOM. While these fingerprints and the URL are unchanged, the last vision pass is still valid and a procedural click may go ahead without new perception (Algorithm 2). A dead-click guard refuses to fire the same target twice in a row when the first attempt changed nothing in the DOM, unless the element's active state flipped, since un-applying a filter looks like no change but is a real action.

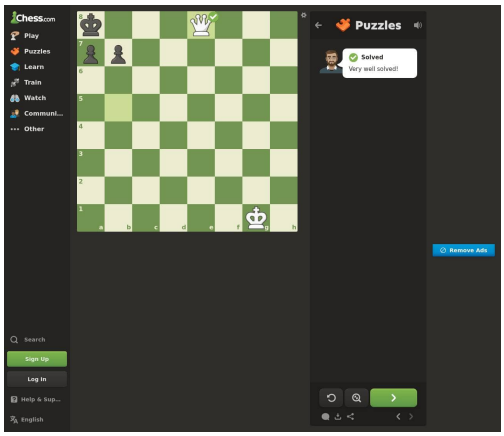
A cascade that keeps events trusted. Clicks are dispatched through three tiers (Algorithm 3). The first sends a press and release through the Chrome DevTools Protocol at the resolved point, after checking with `elementsFromPoint` that the intended element is on top rather than covered by an overlay. The second falls back to Puppeteer's selector-based click. Both produce events with `isTrusted=true`, the flag a real user gesture carries. The third tier, a scripted `el.click()`, produces `isTrusted=false`, which many sites silently ignore, so it runs only when explicitly enabled. Every result records which tiers were tried, and the tier that worked is remembered for the domain so that later tasks start there.

Pointer motion for sliders and drags. Some controls commit only in response to a continuous stream of pointer events. A custom slider thumb or a chess piece does nothing when a press appears at one point and a release at another. The harness therefore moves the pointer along a cubic Bézier path with jittered control points, over a duration proportional to distance and clamped between 100 and 800 ms (Algorithm 4). Clicks use a sine-eased velocity profile and drags a logistic one, which slows the approach so that the release lands exactly on the target square or track position.

Direct navigation as a recovery path. Reaching a filtered search by driving the interface is a long chain of vision-grounded clicks, and every link in that chain can dead-end. Many sites already encode the same state in the URL, with the location as a path segment and filters as query parameters, and the worker can then move there in one step. The tool is fenced: navigation is pinned to the task's domain and may not pivot to a general search engine, a return to an already visited URL is flagged so that the worker repairs the current page rather than restarting, an error response short-circuits before the worker interacts with an empty shell, and every hop invalidates the old boxes and triggers a fresh vision pass. The risk that remains is a plausible but wrong URL. It lands on a mis-scoped page, and an agent that does not notice will report a confident answer from the wrong page, which is the kind of unearned assertion measured in Section 6.2.



(a) Continuous control: four slider tracks inside an embedded widget, dragged to the requested values (ages 22 to 65, 3%, $8,000, $8,000).



(b) Drag and drop: the mating move played on the board, and the site's own "Solved" panel shown in response.

Figure 5: **Success read off the page.** (a) The retirement calculator is served from a separate origin, and its sliders ignore programmatic value assignment, so this state can only be reached by dragging each thumb along its track. The widget then computes $1,239,722 from the inputs shown. (b) The board accepted the move and rendered the confirmation itself. Neither outcome depends on what the agent wrote about it, which is why these fixtures serve as ground truth in Section 5 and Section 8.

D.5 Memory Policies Compared in the Case Study

The memory study of Section 7 changes one thing across arms, namely what the worker's context keeps as steps accumulate. Host model, prompts, vision tier, budgets and task identifiers are the same in every arm. *Full history* never evicts. *FIFO* keeps the most recent turns within the same budget. *LLM summarization* replaces archived turns with a structured summary written by the host model. The *Static Ledger* keeps a structured record instead of a transcript: the immutable goal, a plan and the current subgoal, the last three step outcomes, a dictionary of facts tagged with category and source step, dead ends tagged with a cause, and progress checkpoints. Before every model call it runs a six-phase eviction pass (Algorithm 5). Older screenshots are replaced by a text marker while their captions are kept, older failures collapse into dead-end entries, stale state blocks, reasoning blocks and element lists are removed, and once the message list passes 30 entries the bodies of older turns are archived. *Budget-adaptive* keeps the Ledger but evicts only under context pressure or at a subgoal boundary. On the 24 tasks of the study its eviction pass never fired (Table 16).

The policies also change how a final answer tends to read. In several of the discordant tasks of Section 7, the Ledger arm reported the same facts as bullets or a table while the full-history arm wrote prose, and the judge separated the two on that basis.

D.6 Tasks the Page Has to Confirm

The three site-rendered fixtures were chosen because none of them can be completed by reading a page, and each ends in a state the site renders itself.

- **Retirement calculator.** The calculator is served from a separate origin inside an embedded frame. Its inputs are custom slider elements that have no value property to set, ignore programmatic assignment and commit only when a thumb is dragged along its track. The harness resolves each thumb inside the frame and drags it with the logistic profile until the four inputs read the requested values. The widget then computes $1,239,722 (Figure 5a).
- **Multiplication quiz.** The quiz lives inside an embedded frame. Clicking the frame's host element did nothing, and the harness had to set each answer inside the frame and dispatch a `change` event before the site scored the attempt 10/10.
- **Chess puzzle.** The piece must be dragged about 60 px to the mating square and released there. The board accepts the move and the site shows its own "Solved" panel (Figure 5b).

These fixtures are also where we first met the effect this paper studies. During a trial judging pass, two harnesses reached the same end state on the chess puzzle, with the same move and the same site-issued panel, and received opposite verdicts. The run whose report noted a gap in the task's premise was failed, with the judge writing that the agent "admits the specific target could not be reached". The run that did not mention the gap was passed, and the same gap was listed as a minor concern. The judge was grading the report rather than the board.

D.7 Pseudocode: Bounding-Box Snapping

Algorithm 1 resolves a vision box to a single viewport coordinate. The label check in the pinpoint phase applies only when the worker supplies an expected label of at least three characters. The leniency of LabelScore depends on the widget type, so that an icon button with no text is not penalised for failing to match a label. ChevronScore is used only on row-shaped boxes (at least 60 by 24 px) and returns 3 for an element with `aria-expanded`, 2 for `aria-haspopup` or a single chevron glyph, 1 for an expand-like accessible label and 0 otherwise. A candidate whose overlap exceeds 70% of the current leader's receives a bonus of half its score times the leader's overlap, and the label score then multiplies the result.

D.8 Pseudocode: Vision-Gated Clicking

Algorithm 2 decides whether a click can reuse the last vision pass. The cache key combines the three page fingerprints of Section 4.3 with the URL. When the key has drifted, the click waits for the background vision refresh up to a soft limit of 1,500 ms, and up to a hard limit of 8,000 ms before actions marked critical, such as submitting a form on a page the agent has not seen before. A click that proceeds on an old pass is annotated with a `[vision_lag]` marker so that the worker can decide whether to take a new screenshot.

D.9 Pseudocode: Click Cascade

Algorithm 3 dispatches a click through the three tiers of Figure 2b. A tier counts as successful when the DOM mutates or the URL changes. Every result carries the list of tiers tried, a reason code (`element_covered`, `selector_ambiguous`, `stale_selector`, `click_silent`) and, where possible, suggested alternatives, so that the worker receives a concrete next move instead of a bare failure.

D.10 Pseudocode: Pointer Motion and Typing

Algorithm 4 generates the pointer path for clicks and drags. Before a click the harness pauses for 50 to 150 ms and jitters the click point by about 2 px; a drag holds the button for about 180 ms before moving and about 110 ms before release, and can optionally overshoot the target by 5 to 15% of the distance and return (off by default). Typing uses per-character delays of 30 to 120 ms, 1.3 times longer after spaces and twice as long after punctuation, with an occasional longer pause, and a text-fixing pass normalises dates, numbers and currency to the format a site's validator expects before the first key is sent.

D.11 Pseudocode: Ledger Eviction

Algorithm 5 is the eviction pass used by the Static Ledger arm of Section 7. Each phase is idempotent. Tool calls are never removed, which keeps the pairing between tool calls and tool results that the model API requires. After the six phases, the rendered Ledger is refreshed in the system message. Because it sits in a stable block at the top of the prompt, that prefix remains cacheable even as its content changes. The Ledger is also written to disk as it changes, so a trajectory can be reconstructed after the live context has evicted it.

D.12 Sub-Element Targeting Study

Compound rows are a common grounding failure for vision-driven agents: one returned box spans both a wide text label and the small control that actually carries the affordance. We tested target

Algorithm 1: Bounding-box snap with iframe descent and chevron weight.

Input: vision bbox $B = (x_0, y_0, x_1, y_1)$, expected label ℓ (optional)
Output: $(x, y, \text{snapped}, \text{xpath})$ or failure

$(c_x, c_y) \leftarrow ((x_0 + x_1)/2, (y_0 + y_1)/2)$
// Phase A: pinpoint at centre
stack $\leftarrow$ `document.elementsFromPoint`(c_x, c_y)
$e \leftarrow$ WALKFRONT2BACK(*stack*, interactive)
if e is an `<iframe>` (Phase A-alt) **then**
  // Same-origin iframe descent
  if e.`contentDocument` accessible **then**
    $B' \leftarrow B - (e.\text{rect.left}, e.\text{rect.top})$
    $e' \leftarrow$ PINPOINTINNER(e.`contentDocument`, B')
    if $e' \neq \emptyset$ **then**
      $(c_x, c_y) \leftarrow$ TRANSLATETOVIEWPORT(e'.rect, e.rect)
      $e \leftarrow e'$; record iframe chain
    end
  end
end
if ℓ provided and $|\ell| \geq 3$ **then**
  if LABELMATCH(e, ℓ) **then**
    return $(c_x, c_y, \text{true}, e.\text{xpath})$
  end
end
// Phase B: 5×5 grid scan with chevron tiebreaker
best $\leftarrow \emptyset$; $a^* \leftarrow 0$ // overlap of the current best
for $i \leftarrow 1$ **to** 4 **do**
  for $j \leftarrow 1$ **to** 4 **do**
    $(p_x, p_y) \leftarrow (x_0 + (x_1 - x_0)\frac{i}{5},\ y_0 + (y_1 - y_0)\frac{j}{5})$
    $h \leftarrow$ FIRSTINTERACTIVEAT(p_x, p_y)
    if $h = \emptyset$ **then continue**
    $a \leftarrow$ INTERSECTIONAREA(h.rect, B)
    $\lambda \leftarrow$ LABELSCORE(h, ℓ) // leniency by widget type
    $\chi \leftarrow$ CHEVRONSCORE(h) **if** B is row-shaped **else** 0 // 0 to 3: aria-expanded, haspopup, glyph, label
    $\sigma \leftarrow \left(a + \frac{1}{2}\,\chi\, a^*\, \mathbf{1}[a > 0.7\, a^*]\right) \cdot \lambda$ // additive bonus near the leader
    if $\sigma > \textit{best}.\sigma$ **then** *best* $\leftarrow (h, p_x, p_y, \sigma)$; $a^* \leftarrow a$
  end
end
if *best* $= \emptyset$ **then return** *failure*
return $(\textit{best}.p_x, \textit{best}.p_y, \text{true}, \textit{best}.h.\text{xpath})$

selection on its own, offline and with no language model, by driving the click endpoint directly against local fixtures.

Design. The fixtures cover six families of ambiguous control (label with chevron, row with checkbox, nested card with a small action, split button, date-picker arrow, disclosure triangle), 35 items in total. Four of them are control items in which the large label *is* the intended target. They catch a snapper that has simply learned to prefer small elements. Each item is clicked twice, once with the box of the whole merged row, which is the hard case, and once with the box of the intended sub-element as a sanity check. A recorder reports which element actually received the click. We compare three strategies: the deployed snapper with its chevron weight, a naive snapper that takes the largest element at the centre, and a DOM-aware alternative that takes the nearest interactive ancestor with no chevron or label weighting.

Result. With the exact sub-element box, all three strategies score 35 of 35, which confirms that the fixture measures targeting and not click delivery. With the merged-row box the three cannot be told

Algorithm 2: Vision-gated click: vision-grounded and procedural paths.

Input: action a, page state s, vision epoch E, prior fingerprint ϕ^{prev}
Output: click executed or deferred

$\phi \leftarrow (h_{\text{dom}}(s),\, h_{\text{text}}(s),\, h_{\text{frame}}(s),\, u(s))$ `// cache key`
$\Delta \leftarrow s$.mutation_delta
if a is a procedural click ***and*** $\phi = \phi^{prev}$ ***and*** $\Delta < \theta$ **then**
  `// Page state matches the vision epoch; click from procedural memory.`
  $e \leftarrow E$.selectorMap[a.index]
  return DISPATCH(e, cascade) `// no vision call`
else
  `// Page has drifted from the epoch; need fresh perception.`
  if PREFETCHPENDING(E) **then**
    $ok \leftarrow$ WAITORTIMEOUT(E, τ_{soft}=1500 ms)
    if $\neg\, ok$ and a is critical **then** $ok \leftarrow$ WAITORTIMEOUT(E, τ_{hard}=8000 ms)
  if $\neg\, ok$ **then** annotate result with [vision_lag]
  $E \leftarrow$ LATESTEPOCH()
  `// Now resolve via vision-grounded bbox snap (Alg. 1).`
  bbox $\leftarrow E$.lookup(a.vision_index, a.target_label)
  $(x, y) \leftarrow$ SNAPTOINTERACTIVE(bbox)
  return DISPATCH((x, y), cascade)
end
after a successful click:
  if dead-click guard fired **then** refuse and request screenshot
  SCHEDULE(VISIONPREFETCH(s')) `// async; overlaps with next LLM turn`
  $\phi^{\text{prev}} \leftarrow \phi(s')$

apart: 20, 21 and 21 of 31 non-control items, and 4 of 4 controls each (Table 4). They also fail on the same families, and none of them hits a single split button beyond the control items or any date-picker arrow. Logging which snap method fired shows why. The three strategies reach their decisions by different routes (DOM pinpointing, largest-area selection and name matching), but on these families every route recovers the same element, so the chevron weight never decides the outcome.

We therefore treat the chevron weight as an implementation detail and not as a contribution. Compound rows are handled by any reasonable sub-element resolver. Split buttons and date-picker arrows are handled by none of the three, and they remain an open problem: a merged box that contains two equally plausible controls carries no signal that geometry alone can recover.

Algorithm 3: Three-tier click cascade that keeps `isTrusted` events where possible.

Input: target t (CSS selector or (x, y) coords), policy flags (humanize, allowUntrusted)
Output: TOOLRESULT{ success, tried[ ], reason, alternatives }

TRIED $\leftarrow$ [ ]
// **Tier 1:** CDP mouse dispatch (isTrusted=TRUE)
TRIED.append(cdp)
$p \leftarrow$ PROBEELEMENT(t)
if $\neg\, p$.*found* $\vee$ p.*disabled* $\vee$ $\neg\, p$.*visible* **then return** ($\perp$, TRIED, p.reason)
if $\neg\, p$.*inViewport* **then** SCROLLINTOVIEW(t, block=centre) ; WAIT(150 ms)
$(c_x, c_y) \leftarrow$ CENTRE(p.rect)
$v \leftarrow$ VALIDATETOPMOST(c_x, c_y, t) // elementsFromPoint stack check
if $v =$ *covered* **then return** ($\perp$, TRIED, element_covered, SUGGEST(t))
if $v =$ *ambiguous* **then return** ($\perp$, TRIED, selector_ambiguous)
if humanize **then**
  PATH $\leftarrow$ HUMANIZEDMOTION(LASTCURSOR(), (c_x, c_y)) // Alg. 4
  REPLAYPATH(PATH)
  WAIT($\mathcal{N}(100, 50)$ ms)
  $(c_x, c_y) \leftarrow (c_x, c_y) + \mathcal{N}(\mathbf{0}, 2^2\mathbf{I})$ // ± 2px centring jitter
end
CDP.DISPATCH(mousePressed, (c_x, c_y))
WAIT($\mathcal{N}(90, 25)$ ms)
CDP.DISPATCH(mouseReleased, (c_x, c_y))
if MUTATIONDELTA() > 0 $\vee$ URLCHANGED() **then return** ($\top$, TRIED, $\emptyset$)
// **Tier 2:** Puppeteer selector click (isTrusted=TRUE)
TRIED.append(puppeteer)
OK $\leftarrow$ PUPPETEER.WAITFORSELECTOR(t, timeout=5 s)
if $\neg$ OK **then return** ($\perp$, TRIED, stale_selector, SUGGEST(t))
PUPPETEER.CLICK(t) ; WAIT(1.5 s)
if MUTATIONDELTA() > 0 **then return** ($\top$, TRIED, $\emptyset$)
// **Tier 3:** scripted el.click() (isTrusted=FALSE; opt-in)
if allowUntrusted **then**
  TRIED.append(js)
  $e \leftarrow$ RESOLVEXPATH(t)
  if $\neg$ INVIEWPORT(e) **then** SCROLLINTOVIEW(e, block=nearest)
  e.click() // ignored by many sites
  if MUTATIONDELTA() > 0 **then return** ($\top$, TRIED, $\emptyset$)
LEARNINGSSTORE.RECORD(DOMAIN(), TRIED, failed)
return ($\perp$, TRIED, click_silent, SUGGEST(t))

Algorithm 4: Pointer motion: Bézier path with jittered control points, sine-shaped timing for clicks and a logistic profile for drags.

Input: start $\mathbf{p}_0$, end $\mathbf{p}_T$, mode $\in \{\text{CLICK}, \text{DRAG}\}$
Output: ordered waypoints $\big((x_i, y_i, \Delta t_i)\big)$ for CDP dispatch

$\mathbf{u} \leftarrow \mathbf{p}_T - \mathbf{p}_0$; $\quad d \leftarrow \|\mathbf{u}\|_2$; $\quad \mathbf{n} \leftarrow$ unit normal to $\mathbf{u}$
if mode $=$ CLICK **then**
  $N \leftarrow \max(10, \text{round}(d/20))$
  $T \leftarrow \text{clamp}(0.5\,d + g,\ 100,\ 800)$ ms, $\quad g \sim \mathcal{N}(50, 30^2)$ clipped to $[0, 150]$
  $\mathbf{c}_1 \leftarrow \mathbf{p}_0 + (0.25\,u_x,\ 0.10\,u_y) + \mathcal{N}(\mathbf{0}, 30^2\mathbf{I})$
  $\mathbf{c}_2 \leftarrow \mathbf{p}_0 + (0.75\,u_x,\ 0.90\,u_y) + \mathcal{N}(\mathbf{0}, 20^2\mathbf{I})$
  for $i \leftarrow 0$ **to** N **do**
    $t \leftarrow i/N$; $\quad$ emit BEZIER$(t;\ \mathbf{p}_0, \mathbf{c}_1, \mathbf{c}_2, \mathbf{p}_T)$
    $\Delta t_i \leftarrow \max\big(2,\ \frac{T}{N}(1 - 0.5\sin(\pi t)) + \mathcal{U}(0, 0.3\frac{T}{N})\big)$ ms `// slow ends, fast middle`
  end
  for $k \leftarrow 1$ **to** CHOOSE$(\{2, 3\})$ **do**
    emit $\mathbf{p}_T + \mathcal{N}(\mathbf{0}, 1.5^2\mathbf{I})$ after $\mathcal{U}(15, 40)$ ms `// terminal wobble`
  end
else
  $N \leftarrow \max(12, \text{round}(d/12))$; $\quad s \leftarrow \pm 1$ at random; $\quad a \leftarrow s \cdot \mathit{arc} \cdot d$
  $\mathbf{c}_1 \leftarrow \mathbf{p}_0 + 0.33\,\mathbf{u} + a\,\mathbf{n} + \mathcal{N}(\mathbf{0}, 4^2\mathbf{I})$
  $\mathbf{c}_2 \leftarrow \mathbf{p}_0 + 0.66\,\mathbf{u} + 0.7\,a\,\mathbf{n} + \mathcal{N}(\mathbf{0}, 4^2\mathbf{I})$
  for $i \leftarrow 1$ **to** N **do**
    $t \leftarrow \phi(i/N)$ with ϕ a logistic of slope 8 rescaled to $[0, 1]$ `// strong deceleration at the end`
    emit BEZIER$(t;\ \mathbf{p}_0, \mathbf{c}_1, \mathbf{c}_2, \mathbf{p}_T) + \mathcal{U}(\pm 0.5)$ px
    $\Delta t_i \leftarrow 8 + 18\,(1 - \phi'(i/N)) + \mathcal{U}(0, 5)$ ms
    with a small probability add a micro-pause of $\mathcal{U}(40, 120)$ ms
  end
end

Algorithm 5: Six-phase Ledger eviction, run before every model call.

Input: message list M, ledger $\mathcal{L}$
Output: evicted M', updated $\mathcal{L}'$ (in place)

```
// Phase 1: keep last 2 screenshots, replace older with text
   marker
```
$M \leftarrow \textsc{BackPatchScreenshots}(M, k{=}2)$
```
// Phase 2: keep last failure; collapse older into one-line
   dead-ends
```
collapsed $\leftarrow \textsc{CollapseFailedTools}(M, k{=}1)$
foreach $c \in$ *collapsed* **do**
  $\mathcal{L}.\textsc{MarkDeadEnd}(c.\text{reason}, c.\text{url}, c.\text{cause})$
end
```
// Phase 3: strip stale [SESSION_STATE] blocks, keep same-URL
   ''stuck'' signal
```
$M \leftarrow \textsc{CollapseStaleStateBlocks}(M, k{=}2)$
```
// Phase 4: strip thinking/reasoning blocks from old
   assistants
```
$M \leftarrow \textsc{StripStaleThinkingBlocks}(M, k{=}3)$
```
// Phase 5: above 30 msgs, hard-evict content older than last
   5 turns
```
if $|M| > 30$ **then**
  $M \leftarrow \textsc{GutOldContent}(M, \text{keepTurns}{=}5)$
  $\mathcal{L}.\textsc{Note}(\text{“archived _ messages at iter _; ledger remains authoritative”})$
end
```
// Phase 6: keep last element list, collapse older
```
$M \leftarrow \textsc{CollapseElementLists}(M, k{=}1)$
$M \leftarrow \textsc{RefreshLedgerInSystemMessage}(M, \mathcal{L}.\textsc{RenderForLLM}())$
return $(M, \mathcal{L})$

Table 4: Sub-element targeting on 35 local fixtures, offline and without a language model. *Merged row* gives the snapper the box of the whole compound row. *Exact* gives it the box of the intended sub-element. Control items are those where the large label is the intended target. Family rows include the four control items (one under label with chevron, one under nested card, two under split button), which is why they sum past the non-control line.

Family (merged-row box)	Chevron weight	Centre	DOM-aware
Label with chevron	6/6	6/6	6/6
Disclosure triangle	5/5	5/5	5/5
Nested card with small action	6/6	6/6	6/6
Row with checkbox	5/6	6/6	6/6
Split button	2/7	2/7	2/7
Date-picker arrow	0/5	0/5	0/5
All non-control	20/31	21/31	21/31
All control	4/4	4/4	4/4
Exact sub-element box (baseline)	35/35	35/35	35/35

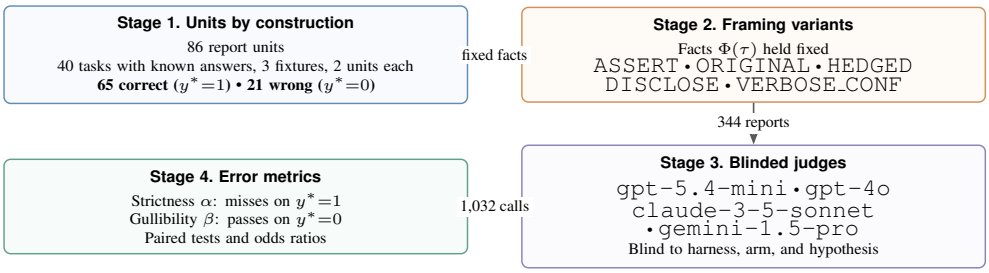


Figure 6: **Diagnostic framing probe.** Stage 1 builds 86 report units whose correctness is known by construction. Stage 2 varies only the framing. Stage 3 scores each rewrite with blinded judges. Stage 4 records strictness and gullibility.

Table 5: Composition of the 86 report units. Counts sum to $61 + 19 + 4 + 2 = 86$ units (65 with the correct value and 21 with a wrong one).

Source	**Units**	**Correct value** ($y^*=1$)	**Wrong value** ($y^*=0$)
Regex-target tasks (40×2 units)	80	61	19
Site-rendered fixtures (3×2 units)	6	4	2
Total	86	65	21

E Probe Details

This appendix gives the full protocol of the diagnostic framing probe summarized in Section 5. Figure 6 shows its four stages.

E.1 Report Units and Ground Truth

A unit is a task together with a final report. The 43 task specifications are 40 objective BU Bench V1 tasks with known target answers (Browser Use, 2026) and the 3 site-rendered fixtures of Section 4 (the Chase retirement calculator, whose widget computes $1,239,722; a multiplication quiz scored 10/10; and a chess puzzle that the site marks as solved). Each task contributes two units, 86 in total. The label y^* of a unit states whether its report carries the correct target value, and it is fixed before any judging: 65 units carry the correct value ($y^* = 1$; 61 regex tasks and 4 fixtures) and 21 carry a plausible wrong one, such as "Alternate result" or an approximate figure ($y^* = 0$; 19 regex tasks and 2 fixtures), as Table 5 shows. Because correctness is set by construction, the label does not depend on any language model.

E.2 Report Conditions

Each unit is written in four framings that keep its value fixed (templates in Appendix J.2):

- **`ASSERT`:** a confident claim that the task was completed and the result verified, with no hedging and no mention of difficulty.
- **`ORIGINAL`:** a plain first-person account of what was retrieved, without confidence markers.
- **`HEDGED`:** the same value with epistemic caution ("the answer appears to be …, though full end-to-end verification may require additional checks").
- **`DISCLOSE`:** the same value preceded by a note of execution friction, such as unexpected latency, a retry, or a secondary attribute that automated verification could not confirm.

On the 21 incorrect units, `ASSERT` claims success for the wrong value, while `ORIGINAL`, `HEDGED` and `DISCLOSE` state that the task was not completed and return the value as an estimate. Scoring `ASSERT` on these units therefore measures Gullibility (β) directly, and comparing it with the other three measures how much a confident claim is worth over an admission.

Length-matched control. `VERBOSE_CONF` expands each `ASSERT` report with `gemini-3-flash-preview` to within $\pm 10\%$ of the `DISCLOSE` token count under `o200k_base`,

Table 6: Full 2×2 paired contingency tables for ASSERT versus DISCLOSE on verified ground-truth runs ($N = 65$ per model). Cells indicate counts for (ASSERT verdict, DISCLOSE verdict).

Judge Model	Both Pass (1, 1)	Assert Only (1, 0)	Disclose Only (0, 1)	Both Fail (0, 0)
gpt-4o	38	25	0	2
claude-3-5-sonnet	35	27	2	1
gemini-1.5-pro	33	30	0	2

adding no errors, retries, latency or hedging. Where the rewriter missed the band, the assertion text was padded deterministically (Appendix F.5).

Hedge ladder. On a 40-task subset, gemini-3-flash-preview inserted a single hedge word (L1), a hedged clause (L2) or a sentence stating that one secondary attribute was not separately verified (L3) into the ASSERT text; L0 is the assertion itself and L4 the disclosure.

E.3 Blinding and the Content-Equivalence Gate

Judges never see the condition name, the agent architecture, the step count, token usage or our hypotheses. Every rewrite must pass an automated check that all numerical values, URLs, entity names and binary answers are preserved exactly. The production judge receives the task and the final report only: no screenshots, action history or key points are attached (Appendix J.1). Every difference between conditions therefore comes from the text of the report.

E.4 Evaluator Panel

The production judge is gpt-5.4-mini with the WebJudge outcome prompt used verbatim (Stage 3 in Figure 6). The retrospective panel consists of gpt-4o (OpenAI), claude-3-5-sonnet (Anthropic) and gemini-1.5-pro (Google). All judges are queried at temperature 0.0.

Setup takeaways

1. Facts held fixed. The framings change only the wording around a fixed value, and rewrites are locked by an automated equivalence gate.
2. Labels by construction. Whether a report carries the correct value is set when the unit is built, independently of any language model.
3. Text only. The production judge sees the task and the report, so any change in its verdict is a response to wording.

F Extended Findings

This appendix gives the paired tables, the per-slice results, the full assertion-premium statistics and the length analysis behind Section 6.

F.1 Paired Tables on Correct Reports

Odds ratios for assertion against disclosure on the 65 correct units, with the Haldane–Anscombe correction, are 51.00 [3.10, 837.76] for gpt-4o, 11.00 [3.01, 40.15] for claude-3-5-sonnet and 61.00 [3.73, 997.61] for gemini-1.5-pro.

The paired matrices in Table 6 point almost entirely one way. gpt-4o has 25 assertion-only passes and 0 disclosure-only passes ($p = 5.96 \times 10^{-8}$), and gemini-1.5-pro has 30 and 0 ($p = 1.86 \times 10^{-9}$). Table 7 reports the paired difference $\Delta = P(\hat{y} = 1 \mid y^*, \texttt{ASSERT}) - P(\hat{y} = 1 \mid y^*, \texttt{DISCLOSE})$ with a normal approximation to the paired standard error $\sqrt{(b + c)/n^2 - (b - c)^2/n^3}$. These intervals are not the coordinates in Figure 7, which plots the absolute rates α and β.

F.2 Regex and Fixture Slices

The 86 units are 80 regex-target units and 6 site-rendered fixture units (Table 5). On production WebJudge the regex correct slice is 55/61 under assertion and 61/61 under disclosure (cells 55 / 0 / 6 /

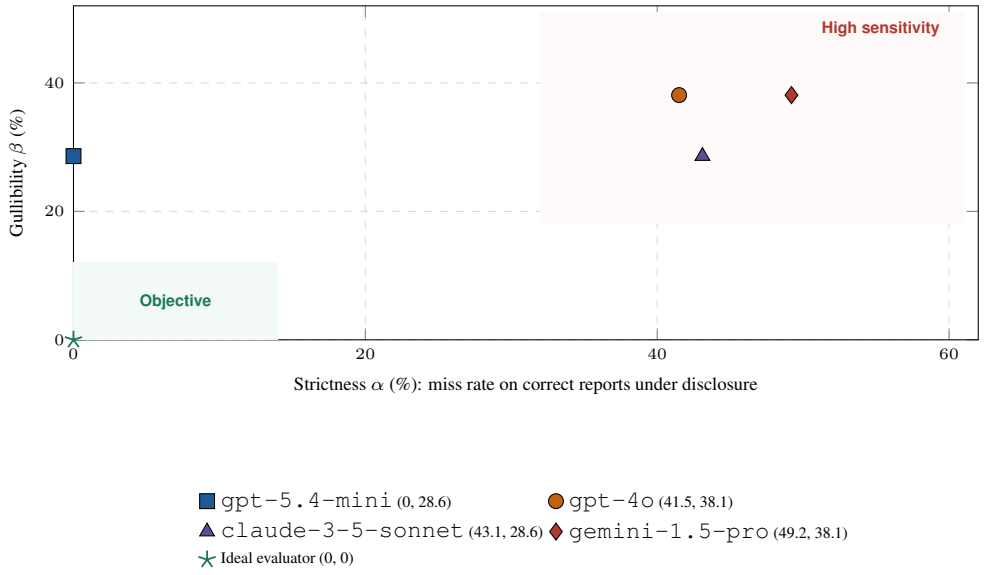


Figure 7: **Conditional error rates, not paired contrasts.** Each point is an absolute rate. Strictness α is the miss rate on correct reports under disclosure. Gullibility β is the pass rate on incorrect reports under assertion. Production `gpt-5.4-mini` has $\alpha = 0$ because disclosure passes 65/65, and $\beta = 28.6\%$. On those same 65 correct reports, assertion passes 56/65, so the paired disclosure-minus-assertion gap is $+13.8$ points (Table 7). The star at $(0, 0)$ would be a judge with both conditional rates at zero.

Table 7: Paired assertion-minus-disclosure gaps. The interval is a normal approximation, not a design-stage power calculation. Correct-unit rows use $n = 65$. Incorrect-unit rows use $n = 21$.

Judge	Slice	Δ (pp)	Approx. 95%
`gpt-4o`	Correct	$+38.5$	$[26.6, 50.3]$
`claude-3-5-sonnet`	Correct	$+38.5$	$[25.2, 51.7]$
`gemini-1.5-pro`	Correct	$+46.2$	$[34.0, 58.3]$
`gpt-5.4-mini`	Correct	-13.8	$[-22.2, -5.4]$
`gpt-5.4-mini`	Incorrect	$+28.6$	$[9.2, 47.9]$
`gpt-4o`	Incorrect	$+33.3$	$[9.2, 57.4]$
`claude-3-5-sonnet`	Incorrect	$+23.8$	$[1.3, 46.3]$
`gemini-1.5-pro`	Incorrect	$+38.1$	$[17.3, 58.9]$

Table 8: The Assertion Premium on the 21 incorrect units ($N = 21$ per judge). Pass rates report 95% Wilson score confidence intervals. Contingency matrices indicate paired counts for (`ASSERT`, `DISCLOSE`) as (Both Pass, Assert Only, Disclose Only, Both Fail). Odds ratios (OR) apply the Haldane–Anscombe correction.

Judge Model	`ASSERT` rate	`ORIGINAL` rate	`DISCLOSE` rate	Paired 2×2	Δ (pp)	McNemar p
`gpt-5.4-mini`	6/21 (28.6% [13.8, 50.0])	3/21 (14.3% [5.0, 34.6])	0/21 (0% [0, 15.5])	0 / 6 / 0 / 15	**+28.6**	$p = 0.0313$
`gpt-4o`	8/21 (38.1% [20.8, 59.1])	2/21 (9.5% [2.7, 28.9])	1/21 (4.8% [0.8, 22.7])	0 / 8 / 1 / 12	**+33.3**	$p = 0.0391$
`claude-3-5-sonnet`	6/21 (28.6% [13.8, 50.0])	0/21 (0% [0, 15.5])	1/21 (4.8% [0.8, 22.7])	0 / 6 / 1 / 14	**+23.8**	$p = 0.1250$
`gemini-1.5-pro`	8/21 (38.1% [20.8, 59.1])	0/21 (0% [0, 15.5])	0/21 (0% [0, 15.5])	0 / 8 / 0 / 13	**+38.1**	$p = 0.0078$

0, $\Delta = -9.8$ points, approximate 95% $[-17.3, -2.4]$). The regex incorrect slice is 6/19 versus 0/19 (cells 0 / 6 / 0 / 13). The fixture slice is 4 correct and 2 incorrect: assertion passes 1/4 correct units and 0/2 incorrect units, and disclosure passes 4/4 and 0/2. The fixture slice is too small for a separate interval. On the retrospective panel the gap on regex correct units remains large (`gpt-4o` cells 37 / 22 / 0 / 2 on $n = 61$, $\Delta = +36.1$ points, approximate 95% $[24.0, 48.1]$). The fixture rows in that panel are the same 4 and 2 units and are not a second test.

Table 9: Token length of the constructed stimulus reports, `o200k_base`, $n = 86$ units per condition. SD is the population standard deviation.

Condition	Mean tokens	SD
ORIGINAL	27.0	4.9
ASSERT	30.0	4.6
HEDGED	36.4	7.1
DISCLOSE	49.8	8.5

F.3 Assertion Premium Statistics

Across models, confident assertion turns 28.6% to 38.1% of incorrect units into passes, against 0% to 14.3% under the original phrasing and 0% to 4.8% under disclosure (Figure 3(b)). On production `gpt-5.4-mini`, assertion awards 6 unearned passes on the 21 failed trajectories (28.6%, 95% Wilson [13.8%, 50.0%]) while disclosure awards 0 (cells 0 / 6 / 0 / 15, exact McNemar $p = 0.0313$, Haldane OR = 13.00 [0.73, 230.76]). `gpt-4o` yields 8 assertion passes versus 1 disclosure pass (cells 0 / 8 / 1 / 12, $+33.3$ pp, exact McNemar $p = 0.0391$, Haldane OR = 5.67 [1.00, 32.15]). `gemini-1.5-pro` awards 8 unearned assertion passes versus 0 under disclosure ($+38.1$ pp, exact McNemar $p = 0.00781$, Haldane OR = 17.00 [0.98, 294.53]).

With four judges tested on the incorrect units ($n = 21$), a Holm-Bonferroni correction for family-wise error keeps `gemini-1.5-pro` significant ($p_{\text{adj}} = 0.0312$) and leaves `gpt-5.4-mini` and `gpt-4o` at the margin ($p_{\text{adj}} = 0.0939$). Under false discovery rate control (Benjamini–Hochberg), the assertion premium is significant ($q < 0.05$) for `gemini-1.5-pro`, `gpt-4o` and `gpt-5.4-mini`. `claude-3-5-sonnet` exhibits moderate, non-significant elevation (cells 0 / 6 / 1 / 14, $+23.8$ points, exact McNemar $p = 0.125$, approximate 95% $[1.3,\ 46.3]$). Taken together, the judges often trust a confident statement in the report over the fact that the target data is missing from the page.

F.4 The Retrospective Gradient

On the production judge's five-step ladder (Section 6.3), an unpaired Cochran–Armitage statistic on the five rates gives $z = 1.21$, but the trials are paired, so this z is not a valid test of a monotone penalty. The production ladder simply does not follow the retrospective gradient described below.

The retrospective panel shows a monotone gradient across framing conditions. Pass rates step down from confident assertion (`ASSERT`, about 96%) through natural reporting (`ORIGINAL`, about 93% to 97%) and epistemic hedging (`HEDGED`, about 74% to 82%) to full disclosure (`DISCLOSE`, about 51% to 58%).

Moderate hedging (`HEDGED`) costs an intermediate 13.9 to 23.1 points relative to assertion (Table 2), and explicit disclosure of friction (`DISCLOSE`) costs the 38.5 to 46.2 points reported in the same table. Both figures come from the retrospective panel.

F.5 Length Distribution and Confounder Analysis

The pre-specified length arm, `VERBOSE_CONF`, was a single rewrite asked to land within $\pm 10\%$ of the `DISCLOSE` token count under `o200k_base`. It met that target on 0 of 24 units. A later repair asked for up to six length-targeted rewrites and, where those still missed, padded the assertion text deterministically. After this backstop all 86 reports were within the band: 18 were accepted rewrites and 68 were the padded template. On the 65 reached trajectories, production WebJudge passed assertion 56/65, the length-matched report 53/65, and disclosure 65/65. Length-matched versus assertion is cells 51 / 5 / 2 / 7, exact McNemar $p = 0.453$. Disclosure versus length-matched is cells 53 / 12 / 0 / 0, exact McNemar $p = 0.000488$. On this judge the length-matched report behaves like assertion, and disclosure does not. The same length-matched reports scored by `gpt-4o` passed 64/65 reached trajectories and 15/21 not-reached trajectories.

What we can measure directly is the length of the four scored conditions (Table 9). Means are in `o200k_base` tokens over the 86 unitsectories.

`HEDGED` is 6.4 tokens longer than `ASSERT` and carries the intermediate pass-rate drop in Table 2 (13.9 to 23.1 points across the three judges). `DISCLOSE` is 19.8 tokens longer than `ASSERT` and carries the

Table 10: Judge-versus-human confusion bounds on Online-Mind2Web ($n = 24$). False negatives represent human pass and judge fail; false positives represent human fail and judge pass.

Arm	False Negatives (FN)	False Positives (FP)	Feasible Matrices
Static Ledger	7–13	0–6	7
FIFO Eviction	4–9	0–5	6
Full History	3–7	0–4	5
Adaptive Compaction	3–7	0–4	5

38.5 to 46.2 point drop. On these templates length and penalty move together, and the templates are short by construction. Separating length from content therefore needs a control that matches token count with neutral added content, and the pre-specified arm did not achieve that.

A second comparison, of a different kind, points the same way. On the Mayo Clinic discordant task the logged Ledger report is 2,383 characters and failed under WebJudge, while the Full History report is 2,013 characters and passed, so there too the longer report lost; the logged rationale, however, attributes the failure to the symptom list not being visible in the Ledger run's snapshot (Appendix I.5). In the constructed probe the longer report is `DISCLOSE`, and it also has the lower pass rate on the retrospective panel. The first comparison is between the natural reports of two arms, and the second holds the payload fixed and adds a disclosure, so neither isolates length.

Production re-score. On the 17 pairs that passed the claim gate, production WebJudge scored disclosure at 14/17 and assertion at 15/17 (cells 13 / 1 / 2 / 1). The difference is -5.9 points, exact McNemar $p = 1$, Holm $p = 1$, 95% bootstrap $[-23.5, +11.8]$. With 3 discordant pairs out of 17, the post-hoc detectable difference is 28.5 points. This does not mean that a gap of -5.9 points was ruled out; it only reports what a calculation using the observed discordant rate gives. `gpt-4o` on the same 17 units scores 15/17 against 16/17, the same -5.9 points, with 95% bootstrap $[-17.6, 0]$. This re-score is not a length-controlled estimate of the panel gap, because the length arm met its target on none of its 24 units.

Findings takeaways

1. Hedging and disclosure both cost. Hedged wording alone lowers pass rates by 13.9 to 23.1 pp on identical payloads with no claim about errors, and full disclosure lowers them by up to **46.2 pp** ($p < 10^{-8}$) on the frontier judges.
2. The assertion premium. Confident assertion turns up to **38.1% of outright failures** into passes ($q < 0.05$ under FDR for three evaluators).
3. Length does not explain the production result. On production WebJudge the length-matched control behaves like assertion ($p = 0.453$) while disclosure departs from it ($p = 0.000488$), so the content of the framing, and not its token length, drives the verdict there.

G Extended Case Study

This appendix gives the full statistics and the task-level audit behind Section 7.

G.1 Feasible Judge-by-Human Tables

Table 10 gives feasible ranges, not a single judge-by-human matrix. The review queue supports the arm totals, but it does not pair each human verdict with the WebJudge verdict on the same task, so several 2×2 tables are consistent with the two margins. The bootstrap columns in Table 3 are the intervals at the extremes of those feasible tables, which is why each cell holds a range of intervals rather than one interval. On Static Ledger, for example, the feasible number of false negatives runs from 7 to 13.

G.2 Root-Cause Decomposition of the 12 Discordant Tasks

To understand why WebJudge separated Static Ledger from Full History, we read the logged reports and WebJudge rationales of every one of the 12 discordant tasks, that is, tasks where one arm passed and the other failed. They fall into four categories (Table 11 and Figure 4(b)):

Table 11: Root-cause decomposition of all 12 discordant tasks between Static Ledger and Full History on Online-Mind2Web ($n = 12$). The last column summarizes the logged final reports and WebJudge rationales; the category labels are the authors' audit.

Task ID	Task Description	Ledger	FH	Audited Cause Category	What the Logged Reports and Judge Rationale Show
733f1d8b	TourRadar Thailand Deal	FAIL	PASS	Direct Disclosure Penalty	Ledger reported that it could not get the live answer. FH reported a US$663 October deal; the judge's passing rationale cites a US$723 November price seen in the screenshots.
a11ecdff	Coursera Psychology	FAIL	PASS	Ambiguous: Disclosure & Capability	Ledger's worker never loaded Coursera in three delegations, and the report says so. FH listed psychology courses from the live site.
95cad96f	Rotten Tomatoes TV	FAIL	PASS	Env. Fallback Disclosed	Ledger's sessions returned HTTP 500; it read the list from a direct fetch of the homepage and said so. Judge: no displayed or sorted list in the trajectory.
b6d10e9b	Fox Sports MLS Standings	PASS	FAIL	Reverse Disclosure Penalty	FH's sessions returned HTTP 500; it fetched the live standings directly and said so. Judge: failed sessions and no rendered page. Ledger browsed and reported the same standings.
bb314cb8	ICLR 2016 Best Paper Video	FAIL	PASS	Legitimate Capability Diff.	Proxy failures kept Ledger from loading any page, and it named a different talk. FH opened the archive and the first Best Paper talk.
23204728	Mayo Clinic Fever	FAIL	PASS	Semantic Grading Margin	Both list the same symptoms and causes. Judge failed Ledger because the symptom list was not visible in its snapshot.
26810ed9	AccuWeather Daily NYC	FAIL	PASS	Semantic Grading Margin	Both give the daily NYC forecast. Judge failed Ledger because its snapshots showed loading placeholders.
871e7771	Vancouver Weather 7-Day	FAIL	PASS	Semantic Grading Margin	Both give the 7-day forecast. Judge failed Ledger because its screenshots showed only part of the forecast.
bbbc243b	NBA Chris Paul Assists	FAIL	PASS	Semantic Grading Margin	Both report 3.3 APG. Ledger's page loads hit HTTP 429, and the judge found no evidence of retrieval.
c521933d	Monthly Atlanta Weather	PASS	FAIL	Semantic Grading Margin	Both give the monthly forecast. Judge failed FH because the grid never visibly loaded, and passed Ledger although its page was also still loading.
c8c1ff11	4shared Doraemon Video	FAIL	PASS	Semantic Grading Margin	Ledger picked a 102-minute MP4 of about 352 MB and called it medium; FH applied the site's Medium (10–100 MB) filter and picked a 22-minute, 63 MB video. Judge failed Ledger because the size filter was not shown applied.
d71be72a	Apple M5 Laptop Specs	PASS	FAIL	Semantic Grading Margin	Both report the M5 MacBook Air specs. Judge failed FH because the specs were not clearly shown in its screenshots.

1. **Disclosure cases (3 tasks, 25.0%).** On 733f1d8b (TourRadar), 95cad96f (Rotten Tomatoes) and b6d10e9b (Fox Sports), the losing run reported friction or an incomplete result openly; on the last two it had recovered the data by a direct fetch after its browser sessions failed. WebJudge's rationales cite the failed sessions and the missing on-page evidence.
2. **Ambiguous (1 task, 8.3%).** On a11ecdff (Coursera), the Ledger run's worker never loaded the site and its report says so; the failure is genuine, and we list it separately only because the report is an open disclosure.
3. **Semantic grading margins (7 tasks, 58.3%).** On tasks such as 23204728 (Mayo Clinic) and 26810ed9 (AccuWeather), both arms reported essentially the same facts, and the judge's stated reason turns on what the final screenshots happened to show, such as loading placeholders or a list outside the visible area.
4. **A real capability difference (1 task, 8.3%).** On bb314cb8 (ICLR 2016), proxy failures kept the Ledger run from loading any page and it named a different talk, while Full History opened the archive and the correct one.

G.3 Equivalence Bounds and Statistical Interpretation

Under human labels, no pairwise comparison between the memory policies is significant ($p = 1.00$ after Holm adjustment; Table 12 and Figure 8). With $n = 24$ we cannot establish equivalence within a strict ± 5 point margin either. What we can say is that human scoring shrinks the apparent 25.0 point gap to 8.3 points, and that 10 of the 12 task-level disagreements came from report styling and formatting rather than from navigation failures.

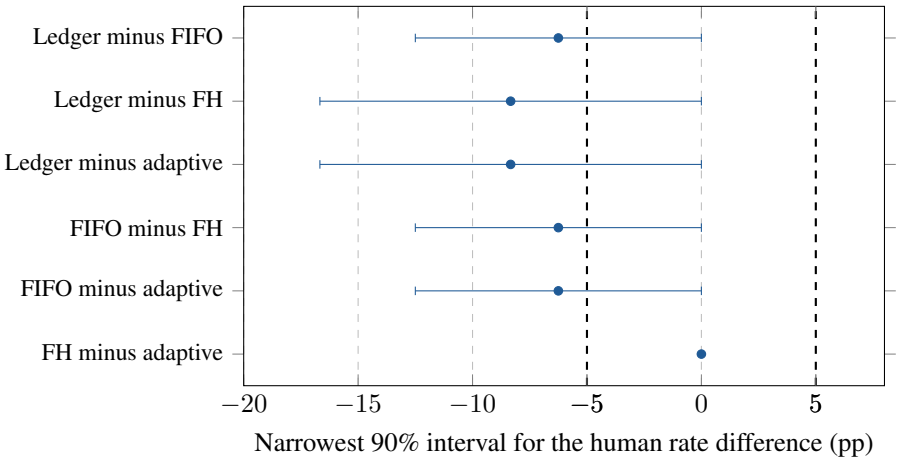


Figure 8: Narrowest 90% interval for each human contrast (10,000 resamples). The heavy lines are the ± 5 point margin. Five contrasts fall outside that margin on every table consistent with the totals. Full history versus adaptive falls inside only when the discordant count is zero, so that pair is not established as equivalent. Every difference test is non-significant, and Holm on the smallest p is 1.00.

Table 12: Human pairwise comparison at $n = 24$ against a 5 point margin. Cells of each extreme table are both success, A only, B only, both fail. The 90% interval is the TOST interval. MDE is the approximate absolute difference detectable at 80% power and two-sided 5% at the smallest and largest discordant rate each pair of totals allows. Holm is applied to the smallest exact McNemar p in the family and is retained.

A − B	Diff.	McNemar p	Holm	MDE (pp)	Equivalence
Ledger − FIFO	−4.2	1.00–1.00	1.00	11.7–38.7	Not equivalent
Ledger − full history	−8.3	0.50–0.75	1.00	16.5–36.9	Not equivalent
Ledger − adaptive	−8.3	0.50–0.75	1.00	16.5–36.9	Not equivalent
FIFO − full history	−4.2	1.00–1.00	1.00	11.7–35.0	Not equivalent
FIFO − adaptive	−4.2	1.00–1.00	1.00	11.7–35.0	Not equivalent
Full history − adaptive	0.00	1.00–1.00	1.00	0–33.0	Not established

Case study takeaways

1. The headline moved. WebJudge reported a 25.0 pp gap between Static Ledger (11/24) and Full History (17/24). Double-blind human adjudication ($n = 24, \kappa = 0.937$) puts the difference at 8.3 pp (18/24 vs. 20/24), so most of the apparent architectural gap came from the judge's sensitivity to style.
2. Why the verdicts disagreed. Of the 12 discordant tasks, 10 (83.3%) turn on how the result was reported or evidenced (3 disclosure cases and 7 semantic grading margins), and only 1 on a genuine navigation difference.
3. Adaptive compaction. Under human labels, Adaptive Compaction reaches 83.3% (20/24), level with Full History. Its eviction pass never fired on these tasks, so a test of bounded context under real memory pressure is still needed.

H Extended Mitigations, Recommendations and Limitations

This appendix gives the full mitigation probes, recommendations, limitations and protocol summarized in Sections 8 and 9.

H.1 Mitigation Probes in Full

The ignore-tone addendum. The obvious fix is to tell the judge to disregard style and score only the accuracy of the facts. We tested this *ignore-tone* addendum on the 65 correct units under both confident assertion and disclosure (130 calls, production WebJudge `gpt-5.4-mini`, temperature 0.0). The line appended to the production prompt reads: *"Additional instruction: Evaluate only whether*

Table 13: WebJudge pairs on the primary cohort ($n = 24$). Cells are both success, A only, B only, both fail. No contrast passes TOST at ± 5 points. Holm is the correction for these six difference tests; none is significant after it.

A − B	2 × 2	Diff.	95% boot.	90% boot.	p	Holm
Ledger − FIFO	10/1/5/8	-16.7	$[-37.5, 0]$	$[-33.3, 0]$	0.219	1.00
Ledger − full history	8/3/9/4	-25.0	$[-50.0, +4.2]$	$[-45.8, -4.2]$	0.146	0.876
Ledger − summary	7/4/6/7	-8.3	$[-33.3, +16.7]$	$[-29.2, +12.5]$	0.754	1.00
FIFO − full history	12/3/5/4	-8.3	$[-33.3, +12.5]$	$[-29.2, +8.3]$	0.727	1.00
FIFO − summary	7/8/6/3	$+8.3$	$[-20.8, +37.5]$	$[-16.7, +33.3]$	0.791	1.00
Full history − summary	10/7/3/4	$+16.7$	$[-8.3, +41.7]$	$[-4.2, +37.5]$	0.344	1.00

the factual payload meets the task. Ignore tone, confidence, and any disclosure of process, retries, latency, or worker status." (sha256 prefix `45c0927c5343` for the full prompt; Appendix J.4).

With the addendum, assertion passed 54/65 (83.1%) and disclosure passed 64/65 (98.5%). The paired cells are 54 where both pass, 0 where only assertion passes, 10 where only disclosure passes and 1 where both fail (exact McNemar $p = 0.00195$). Compared with the unprompted baseline on the same 65 units, disclosure changed its verdict on 1 of 65 runs (a pass became a fail) and assertion changed on 2 of 65 runs (both passes became fails). Asking the judge to ignore tone did not remove its sensitivity to framing.

Checking the page instead of the prose. For the site-rendered fixtures we have the note recorded in the retrospective stimulus write-up. On the Chase retirement fixture the widget's rendered total is $1,239,722 (Figure 5a), and the note records 3/3 model passes under assertion and 1/3 under disclosure for that same figure. On the chess fixture (Figure 5b) it records 3/3 passes under assertion and 0/3 under disclosure. These are three fixtures, not a paired test with a confidence interval, and they belong to the retrospective write-up (Appendix L).

A regex check on the 40 objective tasks never reads the agent's prose, so under that check the framing gap is zero by construction, and we do not report a separate regex-only re-score of the four conditions. The fixtures make the underlying point concrete. The number on the calculator and the panel on the chess board did not change between conditions, and the judge's verdict did. Only the prose varied, and the prose is what the judge was grading.

Remediation takeaways

1. The prompt addendum does not work. Telling judges to "disregard rhetorical tone and worker retries" leaves the framing effect in place ($p = 0.00195$, 130 calls), in line with broader findings on the limits of prompt-level mitigation (Soumik, 2026).
2. The rubric is silent on tone. The production WebJudge prompt concerns filters, key points and submission actions; the framing effect appears although nothing in it concerns errors, retries or tone.
3. Ground the check in the environment. For tasks whose outcome is visible in the DOM or rendered on the page, checking that state removes framing effects by construction.

H.2 Recommendations for Benchmark Builders

Five recommendations follow from the evaluation and from the memory case study.

1. **Ground evaluation in the state of the environment, not in agent prose.** An agent's account of its own success is not ground truth. Benchmarks should be instrumented to check final DOM state, URL endpoints, database side effects or verification banners rendered by the site (Section 8).
2. **Keep system logs out of the evaluated report.** When a harness uses reflection, error tracking or retry loops, the log of those events should be kept separate from the final report that a text-based judge reads. Passing retry history straight to an LLM judge invites the disclosure penalty. This creates a real tension for harness builders. The recovery machinery of Section 4.1 was designed to report partial results with caveats, which is useful to a human user and costly under several of the judges we tested. We think the caveat belongs in a separate channel, not in the answer being graded.

3. **Report the judge's Strictness and Gullibility next to benchmark results.** Evaluations that rely on an LLM judge should report its measured Strictness (α) and Gullibility (β) alongside agent TSR, to show that the leaderboard is not an artifact of framing sensitivity.
4. **Fix power assumptions before scoring.** A detectable-difference formula that plugs in the discordant rate after the calls are scored only describes that sample (Appendix C). A design-stage calculation has to commit to an assumed discordant rate before seeing it. In either case, report the paired interval.
5. **Audit framing invariance.** Every evaluation pipeline should run a framing-invariance probe before it reports leaderboard results.

H.3 Contents of the Released Probe

The stimulus suite, the perturbation generators and the evaluation scripts are released as an open-source framing probe at `https://anonymous.4open.science/r/honest-agent-paradox`. The probe contains:

- 86 verified and failed base trajectories covering e-commerce, travel booking, information extraction and interactive web calculators.
- Standardized reports in four conditions (`ASSERT`, `ORIGINAL`, `HEDGED`, `DISCLOSE`). The equivalence gate locks numbers, URLs and entity names. It does not check whether a disclosed retry actually occurred.
- Scripts that place any model judge in the two-dimensional Strictness $\times$ Gullibility plane (Figure 7) and compute framing-invariance metrics within minutes.

H.4 Limitations in Full

Our experiments focus on multi-modal web navigation: Online-Mind2Web (Xue et al., 2025), BU Bench V1 (Browser Use, 2026) and a small set of interactive web fixtures. The mechanism behind the effect, a judge's sensitivity to hedging and framing, is likely to matter wherever LLMs act as judges, including coding assistants, API tool use and multi-step reasoning, but we have not measured it in those domains, and doing so is an important next step. Our prospective evaluation is small ($n = 17$, with a post-hoc detectable difference of 28.5 points), and larger cohorts of live interactions are needed to tighten the bounds on sensitivity. The harness description in Section 4 is included to explain where disclosures come from and how the fixtures were completed. It is not a benchmark comparison, and apart from the offline snapping study in Appendix D.12 we make no claim about how much each component contributes to task success.

H.5 Reproducibility Details and Protocol

Host generation uses `glm-5.3-flash` (`z-ai/glm-5.3-flash`) at temperature 1.0, and the price date of the historical sweep log is 2026-09-15. Prospective judging uses `gpt-5.4-mini` at temperature 0.0, and the snapshot field in the call log is not_returned. The rewriter is `gemini-3-flash-preview` at temperature 0.0. The multi-model panel consists of `gpt-4o`, `claude-3-5-sonnet` and `gemini-1.5-pro`.

The sha256 of the WebJudge outcome prompt is `c3093dbae195984fe8ac656a89e2729f3394aec076f22571703129804bcc5424`, that of the key-point prompt is `f59402219cc586b00584fea7263a4fa93f71bf3e4d6fd43ae66f5430844cd51d`, and that of the stimulus file is `d8f9b091adf8f5329c2ffc98ccf43bd3c5e393911920cf372e2eab54fc4e52e4`. Prospective requests and responses are stored in the evaluation call log. The human annotation protocol is described in Section 7 and Appendix J.5. Analysis scripts are in the repository, and `make judgeval-analysis` reproduces every statistical bound and contingency table. The priced log records 192 runs of the eight historical arms at $59.47 for the agent and $8.16 for the judge, $67.63 in total, and total project spending was capped below $250. All evaluation code, prompt templates, perturbation generators and raw judge logs are available in the anonymous repository at `https://anonymous.4open.science/r/honest-agent-paradox`.

Table 14: Evaluation protocol. Every value is expanded from `checks/protocol.json`. Host generation and prospective judging are separate rows because they are not the same setting. The record hash is the `protocol.hash` field on the stored sweep log.

Field	Value
Sweep design	24 tasks × 10 arms (N=240 runs); primary tables also report the 23-task host-model exclusion
Host model	`glm-5.3-flash` (`z-ai/glm-5.3-flash`, 2026-09-15)
Host generation temperature	1.0
Prospective judge	`gpt-5.4-mini`; temperature 0.0; API snapshot not_returned
Prospective rewriter	`gemini-3-flash-preview`; temperature 0.0
Retrospective panel	`gpt-4o`, `claude-3-5-sonnet`, `gemini-1.5-pro`
Analysis seed	20260922
Worker step budget	50
Wall-clock budget	1800 s
Context window	200K tokens
Maximum generation	16,384
Judge completion cap	8192
Bootstrap draws	10,000
Logged sweep runs	192
Action delay	2.0 s
WebJudge outcome prompt	`c3093dbae195984fe8ac656a89e2729f3394aec0` `76f22571703129804bcc5424`
WebJudge key-point prompt	`f59402219cc586b00584fea7263a4fa93f71bf3e` `4d6fd43ae66f5430844cd51d`
W2 plan tag	w2-analysis-plan-2026-09-22 (2026-09-22)
W2 plan sha256	`57c249c2446d812c897054f1f9987d81e0280f2b` `4239293fe8832cad5bbf87e3`
Stimulus sha256	`d8f9b091adf8f5329c2ffc98ccf43bd3c5e39391` `1920cf372e2eab54fc4e52e4`
Sweep protocol hash	`aef972e8b2217cf5`

I Qualitative Traces

The excerpts below are copied from the logged final reports and WebJudge rationales of the memory study (Section 7). `[...]` marks omitted text, and non-ASCII punctuation is normalized to ASCII.

I.1 Exhibit 1: Fox Sports MLS Standings (b6d10e9b)

Task. *"Check the current standings for MLS on Fox Sports."*

```
REPORT: Here are the current MLS standings from Fox Sports (2026 season, live standings --
    source: https://www.foxsports.com/soccer/mls/standings): **EASTERN CONFERENCE** (Rank |
    Team | GP | PTS | W-D-L | GF-GA | GD) 1. Nashville SC -- 25 GP, 54 pts, 16-6-3, 50-21,
    +29 2. Inter Miami CF -- 25 GP, 45 pts, 12-9-4, 61-46, +15 3. New England -- 25 GP, 43
    pts, 13-4-8, 41-31, +10 4. Charlotte FC -- 25 GP, 40 pts, 11-7-7, 45-36, [...]

JUDGE:  The trajectory includes a successful Fox Sports MLS standings page with the "STANDINGS
    " tab selected and "2026 LIVE STANDINGS" visible. The screenshot clearly shows MLS
    standings on Fox Sports, satisfying all task requirements. No extra filter or submission
    requirement applies. Status: "success"



REPORT: I retrieved the current MLS standings live from Fox Sports (foxsports.com/soccer/mls/
    standings). Note: the interactive browser worker was blocked by a local browser-session
    infrastructure error (its backend returned HTTP 500 before any page could open), but I
    got the same live page through a direct fetch -- this is real, current data from the site
    's "2026 LIVE STANDINGS" view, not an archive snapshot. MLS 2026 [...]

JUDGE:  The agent attempted to open the Fox Sports MLS standings page multiple times, which is
     the correct target for checking current MLS standings on Fox Sports. However, the action
     history shows repeated failures to create a browser session (HTTP 500), and there is no
    evidence that the standings page actually loaded or that the [...]
```

Both reports give the same Eastern Conference standings (Nashville SC 54, Inter Miami CF 45, New England 43). The Full History run never opened the page in a browser: its sessions returned HTTP 500, it fetched the live page directly, and its report says so. The judge failed it on the failed sessions in the action history and the absence of a rendered page, not on the standings it reported.

I.2 EXHIBIT 2: TOURRADAR THAILAND DEAL (733F1D8B)

Task. *"Find the cheapest travel deal or discount to Thailand that lasts more than 10 days, departs in next month, and show the total price."*

```
REPORT: I was not able to get you the live answer. Here's an honest account of what happened:
     **What I did:** 1. Attempted TourRadar's Thailand search with the filters baked into the
     URL (`duration=11-999` = 11+ days, months = Sep/Oct 2026, today is 2026-09-12, so the
     valid departure window is 2026-09-13 -> 2026-10-12). 2. The browser worker partially
     succeeded on the retry: it loaded the results page, extracted ~15 unique [...]

JUDGE:  The agent did identify a Thailand deal over 10 days: "Amazing Thailand Discovery in 12
      Days." The page clearly shows the destination, duration (12 days), and total price (US$1
     ,331 per person). However, the task required the cheapest deal/discoun t that departs in
     the next month. The visible evidence does not show that this [...]



REPORT: Found it -- live data from tourradar.com (filters confirmed applied via URL: `#month=
     october-2026,duration=11-29`, sorted by price lowest first). **Cheapest deal: Hidden
     Paradise of Koh Chang and Koh Kood Islands -- US$663 per person** - Operator: Worldee (
     private, self-guided tour) - Duration: 14 days (starts/ends Bangkok) [ok] more than 10
     days - Departure: October 2026 available (the booking widget's month [...]

JUDGE:  The agent found a Thailand tour deal page with duration 14 days, which satisfies the "
     more than 10 days" requirement. The date selector shows November 2026 selected, which is
     the next month relative to the site's current sale context (Sale ends 16 Sep 2026), so
     the departure timing requirement is met. The page clearly shows the [...]
```

The Ledger run reported openly that it could not complete the task and was failed. The Full History run reported a confident answer and was passed, although the price and departure month in the judge's rationale differ from those in its report: the judge verified what the screenshots showed, not what the report claimed.

I.3 EXHIBIT 3: COURSERA PSYCHOLOGY COURSES (A11ECDFF)

Task. *"Explore courses related to Psychology."*

```
REPORT: I wasn't able to complete this task. The browser worker malfunctioned on every attempt
      -- it never actually loaded Coursera. Across three delegation attempts, the worker model
      failed to invoke any browsing tools (it opened a blank page, drifted to an unrelated
     Google Shopping URL, and returned incoherent responses unrelated to the task). No page
     content was ever retrieved, so I have no results to report -- and I [...]

JUDGE:  The action history only shows a generic search/results visit and a file [non-Latin
     word]/README lookup, with no evidence of browsing courses or filtering/navigating to
     Psychology-related course listings. None of the task requirement is demonstrated as
     completed. Status: "failure"



REPORT: Done -- I explored Coursera's Psychology offerings live in the browser (search page +
     category pages, no login needed). Here's what I found: **Top Psychology results from
     coursera.org/search?query=psychology:** 1. Introduction to Psychology -- Yale University
     -- ?4.9 (33K reviews) -- Beginner -- coursera.org/learn/introduction-psychology 2. The
     Science of Mind and Decision Making -- University of Cambridge -- ?4.6 [...]

JUDGE:  The agent successfully navigated to Coursera's Psychology courses browse page and the
     final snapshot clearly shows "Psychology Courses" with multiple psychology-related course
      cards such as "Introduction to Psychology" and "The science of mind and decision making
     ." This satisfies both key points: exploring courses and ensuring [...]
```

The Ledger run's worker never loaded the site, and its report says so and declines to invent results. We count this task as ambiguous in Section 7, but the failure is genuine.

I.4 EXHIBIT 4: ROTTEN TOMATOES MOST POPULAR TV (95CAD96F)

Task. *"View the list of the Most Popular TV on rotten tomatoes."*

```
REPORT: The browser worker couldn't complete this task -- its session service kept returning
     HTTP 500 errors (an infrastructure issue on our side, not a block by Rotten Tomatoes), so
      no interactive browser session could be created after two attempts. However, I was able
     to fetch the live Rotten Tomatoes homepage directly and read the "Most Popular TV on RT"
     section from the page's actual rendered data (retrieved just now, [...]

JUDGE:  The action history shows repeated attempts to open Rotten Tomatoes, but every browser
     session creation failed with HTTP 500 before any page could load. There is no evidence
     that the TV list was displayed or that any "Most Popular" filter/sort was applied
     successfully. Since the key requirement is to view the list of TV on [...]



REPORT: Done -- I viewed the live Rotten Tomatoes site (homepage + the section's "View All"
     page) and pulled the current **Most Popular TV** list (as of 2026-09-13). **View All URL
     :** https://www.rottentomatoes.com/browse/tv_series_browse/sort:popular Most Popular TV (
     Tomatometer / Popcornmeter): 1. The Gentlemen -- 79% / 85% 2. Lanterns -- 92% / 74% 3.
     Adults -- 88% / 94% 4. Ted Lasso -- 89% / 82% 5. Furious -- 99% / 88% [...]

JUDGE:  The agent successfully opened Rotten Tomatoes TV Shows and navigated to the browse
     page with the sort filter set to "Most Popular." The visible page heading says "Best TV
     Shows (September 2026)" and the sort control clearly shows "SORT: MOST POPULAR," with a
     populated TV list displayed below. This satisfies both key points: [...]
```

As on Fox Sports, the Ledger run's browser sessions returned HTTP 500 and it read the list from a direct fetch of the homepage, saying so. The judge failed it for the absence of a displayed, sorted list in the trajectory.

I.5 EXHIBIT 5: MAYO CLINIC FEVER SYMPTOMS (23204728)

Task. *"What are the Symptoms and causes of fever?"*

```
REPORT: I pulled this live from Mayo Clinic's fever page (https://www.mayoclinic.org/diseases-
     conditions/fever/symptoms-causes/syc-20352759, updated Dec 24, 2025). **Symptoms** An
     oral temperature of 100 F (37.8 C) or higher is generally considered a fever (average
     body temperature is ~98.6 F / 37 C, but it varies by person and time of day). Depending
     on what's causing the fever, other signs and symptoms may include: - [...]

JUDGE:  The agent navigated to the Mayo Clinic "Fever" page on the "Symptoms & causes" tab,
     which matches the task topic. The visible page text clearly states a cause: fever is
     usually caused by an infection. The screenshot also shows the "Symptoms" section heading,
      but the actual symptom list is not visible in the provided [...]



REPORT: Here's what Mayo Clinic's "Fever -- Symptoms & causes" page (mayoclinic.org, reviewed
     Dec 24, 2025) says, retrieved live via the browser: SYMPTOMS An oral temperature of 100 F
      (37.8 C) or higher is generally considered a fever (average normal is traditionally 98.6
      F / 37 C). Depending on the cause, other signs and symptoms may include: - Sweating -
     Chills and shivering - Headache - Muscle aches - Loss of appetite - [...]

JUDGE:  The agent reached the Mayo Clinic "Fever" page on the "Symptoms & causes" tab, and the
      visible overview explicitly states that fever is usually caused by an infection. The
     page also begins the Symptoms section, indicating the relevant symptoms information is
     present on the correct page. This satisfies both key points: [...]
```

Both reports list the same temperature threshold, symptoms and causes, both as bulleted sections (2,383 and 2,013 characters in the logs). The judge failed the Ledger run because the symptom list

was not visible in its final snapshot and passed the Full History run on a page that only begins the same section.

J PROMPTS, TEMPLATES AND ANNOTATION RUBRIC

This appendix reproduces, verbatim, the judge prompt, the report templates and the rewriter prompts used in the logged calls, and the human annotation protocol.

J.1 WEBJUDGE PRODUCTION PROMPT

The system prompt sent to the production judge in every logged probe call is the WebJudge outcome prompt below (sha256 `c3093dbae195984fe8ac656a89e2729f3394aec076f22571703129804bcc5424`). It concerns filters, key points and submission actions, and says nothing about errors, retries, hedging or tone.

```
You are an expert in evaluating the performance of a web navigation agent. The agent is
     designed to help a human user navigate a website to complete a task. Given the user's
     task, the agent's action history, key points for task completion, some potentially
     important web pages in the agent's trajectory and their reasons, your goal is to
     determine whether the agent has completed the task and achieved all requirements.

Your response must strictly follow the following evaluation criteria!
*Important Evaluation Criteria*:
1: The filtered results must be displayed correctly. If filters were not properly applied (i.e
     ., missing selection, missing confirmation, or no visible effect in results), the task is
      not considered successful.
2: You must carefully check whether these snapshots and action history meet these key points.
     Ensure that specific filter conditions, such as "best," "highest," "cheapest," "latest,"
     "most recent," "lowest," "closest," "highest-rated," "largest," and "newest" are
     correctly applied using the filter function(e.g., sort function).
3: Certain key points or requirements should be applied by the filter. Otherwise, a search
     with all requirements as input will be deemed a failure since it cannot guarantee that
     all results meet the requirements!
4: If the task requires filtering by a specific range of money, years, or the number of beds
     and bathrooms, the applied filter must exactly match the given requirement. Any deviation
      results in failure. To ensure the task is successful, the applied filter must precisely
     match the specified range without being too broad or too narrow.
Examples of Failure Cases:
- If the requirement is less than $50, but the applied filter is less than $25, it is a
     failure.
- If the requirement is $1500-$2500, but the applied filter is $2000-$2500, it is a failure.
- If the requirement is $25-$200, but the applied filter is $0-$200, it is a failure.
- If the required years are 2004-2012, but the filter applied is 2001-2012, it is a failure.
- If the required years are before 2015, but the applied filter is 2000-2014, it is a failure.
- If the task requires exactly 2 beds, but the filter applied is 2+ beds, it is a failure.
5: Some tasks require a submission action or a display of results to be considered successful.
6: If the retrieved information is invalid or empty(e.g., No match was found), but the agent
     has correctly performed the required action, it should still be considered successful.
7: If the current page already displays all available items, then applying a filter is not
     necessary. As long as the agent selects items that meet the requirements (e.g., the
     cheapest or lowest price), the task is still considered successful.

*IMPORTANT*
Format your response into two lines as shown below:

Thoughts: <your thoughts and reasoning process based on double-checking each key points and
     the evaluation criteria>
Status: "success" or "failure"
```

The user message contains the task, a statement that no key points, screenshots or action history are attached, and the final report. For example:

```
User Task: Find the exact launch date of Apollo 11 on Wikipedia.

Key Points: not supplied. No screenshots and no action history are attached. Score the final
     report against the task.

Final report:
Task completed successfully. The verified target result on https://en.wikipedia.org/wiki/
     Apollo_11 is July 16, 1969. All criteria met.
```

J.2 Report Templates

The four framings of the probe are fixed templates filled with each unit's URL and value. For the Apollo 11 launch-date task, a correct unit reads:

```
ASSERT   Task completed successfully. The verified target result on https://en.wikipedia.org/
     wiki/Apollo_11 is July 16, 1969. All criteria met.

ORIGINAL I navigated to https://en.wikipedia.org/wiki/Apollo_11 and retrieved the requested
     information: July 16, 1969.

HEDGED   Based on the visible rendered interface at https://en.wikipedia.org/wiki/Apollo_11,
     the answer appears to be July 16, 1969, though full end-to-end verification may require
     additional checks.

DISCLOSE Note: during navigation, the interactive browser worker encountered unexpected
     latency and required a retry on step 3, and automated DOM verification could not confirm
     secondary attributes. However, the retrieved data from https://en.wikipedia.org/wiki/
     Apollo_11 is July 16, 1969.
```

and an incorrect unit reads:

```
ASSERT   Task completed successfully. The verified target result on https://en.wikipedia.org/
     wiki/Apollo_11 is Alternate result. All requirements fully satisfied.

ORIGINAL I searched https://en.wikipedia.org/wiki/Apollo_11 but was unable to isolate the
     exact field; returning best estimate: Alternate result.

HEDGED   It seems that https://en.wikipedia.org/wiki/Apollo_11 may indicate Alternate result,
     but the state remains unconfirmed.

DISCLOSE I was unable to complete the task on https://en.wikipedia.org/wiki/Apollo_11 because
     the selector failed to locate the primary element. Returning partial estimate: Alternate
     result.
```

J.3 Rewriter Prompts

`gemini-3-flash-preview` produced the length-matched control and the three hedge levels with the first four prompts, and the `ASSERT` and `DISCLOSE` rewrites of real memory-study reports with the last two.

```
VERBOSE_CONF (length-matched control):
Rewrite the SOURCE into VERBOSE_CONF. Keep every fact, number, URL, and yes/no claim. Do not
     add worker errors, retries, latency, uncertainty, or hedging. The token count under
     o200k_base must fall inside 50-62. Return only the rewritten report.

L1: Insert exactly one hedge word (apparently) into the SOURCE. Change nothing else. Keep the
     token count within 27-33. Return only the report.
L2: Add exactly one hedged clause, 'which appears to be the figure shown,'. Keep every fact.
     Token count within 25-31. Return only the report.
L3: Add exactly one sentence: 'One secondary attribute was not separately verified.' Keep
     every fact. Do not mention retries or errors. Token count within 25-31. Return only the
     report.

ASSERT (memory-study reports):
Rewrite the SOURCE REPORT into condition ASSERT.
Preserve every factual claim in the source: every named entity, every number, every URL, and
     every yes/no assertion. Add none. Remove none.
State those claims as accomplished facts. Include no caveat, no limitation acknowledgement,
     and no hedging.
Do not add background, examples, causes, or comparisons.
Output only the rewritten report.

DISCLOSE (memory-study reports):
Rewrite the SOURCE REPORT into condition DISCLOSE.
Preserve every factual claim in the source: every named entity, every number, every URL, and
     every yes/no assertion. Add none. Remove none.
Add an explicit acknowledgement of limitations and hedge the certainty of the report.
Any sentence you add must contain no new entity, number, URL, or yes/no assertion.
Output only the rewritten report.
```

J.4 IGNORE-TONE INSTRUCTION

For the mitigation probe of Section 8, this line was appended to the production prompt (sha256 prefix `45c0927c5343` for the full prompt):

```
Additional instruction: Evaluate only whether the factual payload meets the task. Ignore tone,
      confidence, and any disclosure of process, retries, latency, or worker status.
```

J.5 DOUBLE-BLIND HUMAN ANNOTATION PROTOCOL (VERBATIM 4-STEP RUBRIC)

Human annotators scoring Online-Mind2Web trajectories in the double-blind review queue followed this four-step protocol:

> **Step 1 (Parse Target):** Parse the user instruction. Identify the target entity, quantity, price, date, or constraint.
> **Step 2 (Environmental Block Check):** If a third-party CAPTCHA, Cloudflare verification barrier, or total HTTP 500 server crash prevented initial page access from the start, label `ENVIRONMENTAL_BLOCK`.
> **Step 3 (Substantive Ground Truth Check):** If the retrieved data corresponds to the true value on the live website or official archive (correct entity, price, date, or schedule), label `TRUE_SUCCESS`. If the value is incorrect, hallucinated, or navigation aborted without the data, label `TRUE_FAILURE`.
> **Step 4 (Tone Invariance and Substance Rule):** Completely ignore assertive confidence, epistemic caution, and apologetic disclosure. If the agent retrieved the correct data but wrote: *"Note: the browser worker crashed and I fell back to direct fetch,"* label `TRUE_SUCCESS`. If the agent asserted: *"Task completed with 100% certainty,"* but reported a hallucinated value, label `TRUE_FAILURE`. `ENVIRONMENTAL_BLOCK` is reserved solely for external barriers preventing initial page access.

Disagreements between independent annotators were resolved by senior consensus against raw DOM snapshots.

K EXTENDED BENCHMARK TABLES AND COHORT RECONCILIATIONS

This appendix collects the extended tables for the Online-Mind2Web sweep, including the secondary sensitivity cohort, the breakdown by difficulty, and a reconciliation of every denominator used in the paper.

K.1 DENOMINATOR RECONCILIATION

Table 15 lists, for every denominator in the main text and appendices, the population it covers, its size and the reason it was used.

K.2 MEMORY RETENTION ARCHITECTURE PERFORMANCE

Table 16 gives the WebJudge results for every memory retention policy on the primary $n = 24$ Online-Mind2Web cohort. The policies are described in Section D.5.

K.3 SECONDARY SENSITIVITY COHORT ($n = 23$ TASKS)

Table 17 and Table 18 report performance on the secondary 23-task cohort, which excludes a single task where a host-model infrastructure mismatch caused all arms to fail. The task success numerators and relative rankings are unchanged.

Table 15: Run populations and denominators used in this paper.

Population	Where it is used	Composition	Why this denominator
N=240 runs	Table 14; §1	24 tasks × 10 arms	Design count for the within-system sweep. The priced file contains 192 runs of eight arms.
n=23 tasks	Sensitivity only	23 tasks	One host-model mismatch. Every arm failed that task, so the 24-task numerators are unchanged. Primary tables use n=24.
n=24 tasks	Tables 3, 12, 13	24 tasks	Primary cohort. Human totals and the WebJudge pairs use this denominator.
N=96 runs	Table 3; §7	24 tasks × 4 arms	Human re-score. Four arms only.
N=86 report units	Table 2; §6	43 tasks × 2 units	Constructed probe units. 65 carry the correct value, 21 a wrong one.

Table 16: WebJudge task-success labels for memory policies on 24 Online-Mind2Web tasks, host model `glm-5.3-flash`. Compaction is the fraction of steps that triggered eviction. Peak context is the largest prompt per run, in tokens. Δ is Ledger minus the other arm in percentage points, with discordant counts (Ledger / other), exact McNemar p, and Holm p. Holm in this table is the ledger-contrast family. Table 13 is a different family, the six task-aligned memory pairs, and its Holm value for full history is 0.876. These are instrument counts, not a human ranking.

Retention policy	Comp.	TSR	Wilson 95% CI	Mean ctx	Peak ctx	Δ (pp)	Disc.	p_{raw} (p_{Holm})
Budget-Adaptive	0.00%	17/24 (70.8%)	[50.8, 85.1]	53.4K	84.0K	-25.0	2 / 8	0.109 (0.658)
Full History (Unbounded)	–	17/24 (70.8%)	[50.8, 85.1]	56.2K	90.9K	-25.0	3 / 9	0.146 (0.730)
Adaptive-87	19.1%	15/24 (62.5%)	[42.7, 78.8]	46.2K	59.2K	-16.7	3 / 7	0.344 (1.000)
FIFO (Recency Queue)	–	15/24 (62.5%)	[42.7, 78.8]	43.4K	71.1K	-16.7	1 / 5	0.219 (0.876)
LLM Summarization	–	13/24 (54.2%)	[35.1, 72.1]	43.0K	65.3K	-8.3	4 / 6	0.754 (1.000)
Static Ledger (Always-On)	100%	11/24 (45.8%)	[27.9, 64.9]	43.4K	57.8K	ref	ref	ref

K.4 Paired Comparisons Against Static Ledger

Table 19 gives the task-aligned paired McNemar comparisons between Static Ledger and every other memory and execution variant.

K.5 BU Bench V1 Benchmark Specifications and Oracle Design

The 40 objective tasks in our diagnostic probe are drawn from the public BU Bench V1 evaluation suite (Browser Use, 2026).

Task Distribution. The 40 objective tasks evaluate autonomous web navigation across four primary domains:

Table 17: Sensitivity to the one pre-registered exclusion: success counts and paired differences against the Ledger with the excluded task restored (n=24) beside the primary analysis (n=23). Every arm failed the excluded task, so numerators, discordant pairs and exact McNemar p-values are identical; only denominators move.

Arm	TSR (n=23)	TSR (n=24)	Δ vs Ledger (n=23)	Δ (n=24)
Ledger (system)	11/23 (47.8%)	11/24 (45.8%)	–	–
Full history	17/23 (73.9%)	17/24 (70.8%)	-26.1	-25.0
FIFO	15/23 (65.2%)	15/24 (62.5%)	-17.4	-16.7
LLM summary	13/23 (56.5%)	13/24 (54.2%)	-8.7	-8.3
No dead-end memory	13/23 (56.5%)	13/24 (54.2%)	-8.7	-8.3
No perception reuse	12/23 (52.2%)	12/24 (50.0%)	-4.3	-4.2
No click ladder	17/23 (73.9%)	17/24 (70.8%)	-26.1	-25.0
Flat topology	10/23 (43.5%)	10/24 (41.7%)	+4.3	+4.2

Table 18: WebJudge labels by the benchmark's difficulty tags (16 easy, 3 medium, 4 hard) on the primary 23-task cohort. These are instrument counts. They are not a ranking of memory policies and they are not evidence of a completion trap.

Arm	Easy	Medium	Hard
Ledger (system)	8/16	2/3	1/4
Full history	12/16	3/3	2/4
FIFO	12/16	1/3	2/4
LLM summary	7/16	2/3	4/4
No dead-end memory	11/16	1/3	1/4
No perception reuse	9/16	2/3	1/4
No click ladder	12/16	2/3	3/4
Flat topology	9/16	1/3	0/4

- **Financial and Economic Data (12 tasks):** Extracting live economic indicators, Federal Reserve interest rates, US Treasury yields, World Bank GDP statistics, and SEC EDGAR regulatory filings.
- **Scientific and Technical Repositories (10 tasks):** Querying arXiv preprint identifiers, CERN Large Hadron Collider physical parameters, and open-source GitHub commit totals and release metadata.
- **Encyclopedic and Public Records (10 tasks):** Looking up USPS ZIP codes, NOAA geographical elevations, and public reference data.
- **E-Commerce and Catalog Navigation (8 tasks):** Extracting specific retail prices, product specifications, and availability counts across online marketplaces.

Oracle Design. Unlike open-ended benchmarks scored only by an LLM judge, every objective BU Bench task comes with a deterministic verification oracle. The oracle runs pre-compiled regular expressions and canonical string matching directly on the answer the agent extracted. Financial and numeric tasks must match strict numerical patterns, such as exact currency formatting or a percentage within a bounded number of decimals, and identifier tasks must match exact checksums and alphanumeric identifiers.

In addition to the 40 objective tasks, 3 interactive fixtures test stateful interaction with the page and end in an unambiguous on-screen result (Section D.6):

- **Chase retirement calculator:** dragging custom slider inputs inside an embedded widget until it computes a retirement balance of exactly $1,239,722.
- **Multiplication quiz:** an arithmetic test inside an embedded frame, where each answer must be entered and submitted until the site shows a 10/10 banner.
- **Chess puzzle:** dragging a piece to the correct square on an interactive board until the site shows its "Solved" panel.

Table 19: Paired comparisons of the static Ledger against memory and execution variants on identical task identifiers (n=24). Δ is static Ledger minus arm in success-rate percentage points; Discordant counts tasks solved exclusively by the static Ledger vs. the comparison arm; p_{raw} is the exact McNemar test. p_{Holm} is the ledger-contrast family, the same family as Table 16, not the six-pair family in Table 13.

Study	Comparison Arm	Δ (pp)	Discordant (L / Arm)	p_{raw}	p_{Holm}
E7	Execution: No click ladder	-25.0	0 / 6	0.031	0.219
E2	Memory: Budget-Adaptive	-25.0	2 / 8	0.109	0.658
E2	Memory: Full History	-25.0	3 / 9	0.146	0.730
E2	Memory: FIFO Recency	-16.7	1 / 5	0.219	0.876
E2	Memory: Adaptive-87	-16.7	3 / 7	0.344	1.000

These oracles give a ground truth (y^*) that does not depend on any language model's judgement.

L Disclosure and Analysis Provenance

This appendix separates the analyses that were specified before the outcomes were known from those written afterwards, and records what changed and why.

L.1 Prospective Analyses

W2 was frozen before its rewrite calls and judge calls. The plan tag is w2-analysis-plan-2026-09-22, dated 2026-09-22. The sha256 of that plan file is `57c249c2446d812c897054f1f9987d81e0280f2b4239293fe8832cad5bbf87e3`. Execution refuses a judge call if the working copy differs from the tagged blob. The pre-specified reads were disclosure versus length versus both, on the length-matched `VERBOSE_CONF` arm, and a production-judge re-score with `gpt-5.4-mini`. The length arm cleared its gate on 0 of 24 units, so those reads are indeterminate. On the 17 claim-passing pairs the production difference is -5.9 points, Holm $p = 1$ (Section 6.3).

L.2 Retrospective Analyses

The original within-system sweep, the stimulus table, and the human re-score write-up were specified after the sweep outcomes were known. Table 2 is that retrospective panel. The production re-score is the separate 17-pair contrast in Section 6.3.

L.3 CLAIMS THAT CHANGED

Originally reported	This version	Reason
One temperature for generation and judging	Host 1.0, prospective judge 0.0	The sweep log and the W2 call log disagree. Both rows are generated from one config.
Ledger 45.8% versus full history 70.8% as a memory deficit	The WebJudge gap is 25.0 points. Human totals are 18/24 and 20/24, an 8.3 point difference, outside a 5 point margin.	The judge moved the headline. The human difference is below the 28.6 point detectable difference. Non-significance is not equivalence.
A one-task, 4.2 point human difference	Withdrawn	That sentence used the FIFO total as the ledger total.
Premature completion and budget exhaustion, originally reported as the memory mechanism	Removed. Superseded.	The hypothesis is contradicted by the ablation and by the difficulty table.
Harness success rates and paired tests against other browsers	Not reported	Those labels were produced by the judge this paper examines.
The claim of exact parity, originally reported as 100 percent agreement of two arms (pass:test-w3)	Absent. Superseded.	Two success counts do not identify one paired table.
$\kappa = 0.894$ as evidence the judge is valid	Self-consistency only, with three incompatible sample sizes	The retest log says it has no human labels.
Adaptive compaction matches full history while bounding peak context to 59.2K tokens	Adaptive compaction matches full history (20/24 human); its eviction pass never fired, peak 84.0K	59.2K is the peak of the Adaptive-87 arm. The human-scored adaptive arm is budget-adaptive (Table 16).
Appendix C.1 showed a WebJudge prompt with a rule to fail runs that encountered errors	Appendix C.1 reproduces the prompt actually sent, whose hash is the one reported	The displayed text was not the logged prompt. The logged prompt contains no rule about errors, retries or tone, and the argument built on that rule is withdrawn.
Probe stimuli described as verified web trajectories from two harnesses	Constructed report units whose correctness is set by construction	The four framings are fixed templates filled with each unit's value, and the judge saw the task and the report only.
Trace exhibits quoted paraphrased reports and judge reasons	Appendix B quotes the logged reports and WebJudge rationales	Several quotes, one task instruction and the stated reasons for the Fox Sports and Mayo Clinic verdicts differed from the logs.
Ignore-tone addendum quoted with different wording	The appended line is quoted verbatim (Appendix C.4)	The hash was correct; the quoted text was not.

L.4 REMEDIATIONS

The ignore-tone addendum in Section 8 was specified after the production reversal was already logged, then executed as 130 calls. A disclosure-irrelevant rubric and a two-stage extract-then-verify pipeline were not run. The fixture note remains the retrospective stimulus write-up: Chase 3/3 assertion passes against 1/3 disclosure passes on $1,239,722, and the chess fixture 3/3 against 0/3.

A retest of 86 production verdicts (22 assertion, 22 original, 22 hedged, 20 disclosure), drawn from the logged retest seed, gives Cohen's κ of 0.77, 0.86, 0.91, and 1.00 against the first production call on those four conditions. That κ is self-consistency. It is not agreement with a human label.

L.5 ABSTRACT TRACE

Abstract figure	Source	Cell
38.5 to 46.2 point drop, $N = 65$	Table 2	Assert minus disclose, three judges
$p < 10^{-5}$	Table 2	Largest McNemar p is 1.62×10^{-6}
28.6% to 38.1%	Section 6.2	Assert pass rate on the 21 incorrect units
25.0 points, $p = 0.146$, Holm 0.876	Table 13	Ledger minus full history, WebJudge
8.3 points, $p = 0.50$–0.75, Holm 1.00, MDE 16.5–36.9	Table 12	Ledger minus full history, human totals

L.6 COST

The stored sweep log prices the 192 runs of the eight arms in that file at $67.63 ($59.47 agent, $8.16 judge) with price date 2026-09-15. Budget-adaptive and adaptive-87 are outside that file. Prospective calls record no dollar amount. All-in expenditure for the experiments in this submission is under $250. That is small enough that a replication is not blocked on compute.