IS THIS AGENT EVALUABLE? CERTIFYING VERDICTS ABOUT LLM AGENTS UNDER NONDETERMINISM, ORACLE AND COVERAGE OBSTRUCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

A benchmark score is an instrument reading. The instrument has four stages: a policy that samples actions, an environment that carries state, an oracle that renders a verdict, and a task suite somebody chose. We define *evaluability* as the existence of a procedure that settles a claim from the available evidence at a stated error level. We name five ways it fails, and measure each on live agents across three domains with unrelated checker designs.

Four results. Swapping the tasks moves a score 6.2 times more than rerunning them, so a three-seed error bar prices the cheaper axis. Whether reruns settle a task is a fact about the pass rates, not the checker, so a fixed suite settles *less* as agents improve. Judges agree at 0.761 on benchmark obligations but only 0.513 on capabilities that real agents declare. And cheatability belongs to the interface: our own harness leaks every reference answer, making 0.608 of our suite passable with no task competence, none of it reachable by the five browser primitives our agent had. Of nine pre-registered predictions, 3 survive. Where an obstruction binds, the honest output is *undecided* with the reason named, rather than a score.

1 INTRODUCTION

An agent benchmark result is usually one number. One model scores 0.42, another 0.39, and the ordering is read as a fact about the two systems. This paper asks what has to be true for that reading, then measures how often it holds. The question is not whether agent evaluation is noisy, which is known, but what a score licenses. A claim survives only if every stage of the instrument is decisive: the policy sampling actions, the environment carrying state, the oracle rendering a verdict, and the suite somebody assembled. If one stage is not, the number is printed anyway, because nothing in the pipeline can say otherwise. An evaluation should be able to return *undecided*, and say why. So we name five obstructions and measure each.

The body is one chain: each section answers the question the previous one raises and names the appendix subsection expanding it. Sections 5 to 8 take the sampling stage, Section 9 judgement and reach, Section 10 grouping, Section 12 attribution, Section 13 shortcuts, and Section 14 six findings of ours that were artifacts of our own instrument.

2 RELATED WORK

Sampling error in static evaluation is established: error bars, paired designs, reproducibility failures, standardised reporting and design checklists (Miller, 2024; Biderman et al., 2024; Liang et al., 2023; Reuel et al., 2024). All treat a benchmark as independent items scored by a fixed oracle. Agent evaluation breaks both halves: the environment carries state and the oracle is itself a program or a model. We run VisualWebArena (Koh et al., 2024), τ-bench (Yao et al., 2024) and CRMArena-Pro. Judge reliability is normally agreement with humans (Zheng et al., 2023); we ask which obligations need a judge at all. Appendix A.5 places each literature, difficulty modelling and cheating audits included, and the estimator belonging at each stage.

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.

3 WHAT EVALUABILITY MEANS, AND THE FIVE WAYS IT FAILS

Definition 1 (Evaluability)**.** *Fix an agent A, a claim C about it, and an evidence set E of traces. The triple (A, C, E) is* evaluable at level α *if there exists a decision procedure that returns accept or reject for C with error at most α, using only E. Otherwise the correct output is* undecided*.*

Current practice returns a number, never undecided. Five obstructions can stop a triple from being evaluable, each a different stage failing. *O1, nondeterminism.* Identical reruns give different verdicts, so a single trace establishes nothing. We say nondeterminism rather than randomness because Section 5 shows that removing the sampling source does not remove the disagreement. *O2, judgement.* The check is not decisive. Two cases stay apart: *decidability* asks whether a program could decide conformance, *validity* whether the reference is right. *O3, families.* Failures come in families, so an aggregate hides the structure and a random subset mis-estimates the score. *O4, reach.* The evidence never exercises the claimed capability, and a verdict about an untested skill is unsupported however good the oracle. *O5, shortcuts.* A passing verdict is reachable without the capability claimed. O5 differs in kind. O1 to O4 widen an interval or withhold a verdict, weakening a claim. O5 inverts one, so more evidence makes the error more confident. It binds whenever the action space reaches an object determining the verdict, so cheatability belongs to a pair: the benchmark and the interface.

Two quantities carry the paper. Table 8 defines every term and Appendix A.1 gives the estimators. Two need stating here. A score averages over tasks and each task's contribution over reruns, so it could have come out differently two ways: a different sample of tasks (the *task axis*) or the same tasks rerun (the *rerun axis*). A seed error bar reports the second only. A task is *settled* when at least $\gamma = 0.9$ of its posterior pass rate lies outside $[\alpha, 1 - \alpha]$ at $\alpha = 0.1$, so a further run adds nothing. That says how much doubt is left, not whether the agent passed: 8 of 8 and 0 of 8 are equally settled. *Settleability* is the settled share of a suite, a fact about the score rather than the checker. A task *flips* when identical reruns both pass and fail it.

4 SETUP, AND WHAT WE PREDICTED IN ADVANCE

Table 1: The three live domains and the two declared extensions, with the pre-registration's D1 to D3 labels. The three checker designs are unrelated, so a finding reproducing across them is not one harness's artifact. Extensions are reported, never counted toward a prediction.

Domain	Role	What the checker reads	Tasks	k	Runs
VisualWebArena shopping	D1	the live page: strings, DOM state and the final URL	79	8	3792
CRMArena-Pro B2B	D2	an answer string against a reference, falling back to a model that extracts entities	119	8	952
τ-bench retail	D3	the tool-call sequence against a reference database state, in process, no browser	115	8	920
CRMArena-Pro B2C	extension of D2	same templates, different organisation records	118	8	944
τ-bench airline	extension of D3	disjoint tasks, different policy document	50	8	400

The grids. Table 1 lists the domains. Our main grid is VisualWebArena's shopping environment, restricted to the 79 tasks our harness supports end to end. That is the entire eligible population, so there is no sampling rule to justify, and verdicts come from the benchmark's own programmatic checkers. 6 configurations across three model families drive a byte identical scaffold at $k = 8$, giving 3792 runs, all pinned and logged per call, against independent environment instances never shared concurrently (Appendices A.6 and A.7). **The nine predictions, and what makes them ex ante.** Every threshold was fixed before the matching data existed, in a registration file frozen as an annotated tag. All 12 result files feeding a confirmatory prediction were first committed after it, and the file has only ever been appended to, so a moved goalpost would surface as a deletion. The evidence establishes the *order* of events, not their dates (Appendix A.2, Table 9). P9 is the exception: its registry sweep was committed *in* the freeze commit and is why we studied coverage at all, so it is excluded from the correction, 0 of 42 swept agents exposing a machine-checkable skill declaration at discovery.

5 O1, NONDETERMINISM: CHANGING TASKS MOVES THE SCORE MORE THAN RERUNNING THEM

Table 2: Main grid: 79 tasks, $k = 8$, 3792 runs, zero harness failures. Flip is the share of tasks whose reruns disagree, Dec. is settleability. The CI columns are 95 percent bootstrap widths in points: CI_{both} resamples tasks and runs, CI_{run} runs only, which is what a seed-to-seed error bar measures. †Truncates on 43 percent of runs at any budget we could grant, so its *score* is excluded from ranking claims.

Model	Score	Flip	ICC	DEFF	Dec.	CI_{both}	CI_{run}	Ratio
`claude-haiku-4-5`	0.169	0.215	0.703	5.92	0.709	14.7	2.8	5.2
`claude-sonnet-5`	0.261	0.203	0.790	6.53	0.797	17.6	2.8	6.2
`claude-opus-5`	0.253	0.228	0.768	6.38	0.772	17.6	3.2	5.5
`gemini-3.1-pro-preview`	0.301	0.266	0.746	6.22	0.734	18.0	3.5	5.2
`gemini-3.7-flash`	0.305	0.253	0.757	6.30	0.747	18.5	3.5	5.3
`glm-5.2`†	0.229	0.304	0.667	5.66	0.595	16.2	3.5	4.7

The main result. Table 2 gives the grid, and its last three columns carry it. The interval resampling both tasks and runs is 14.7 to 18.5 points wide, against 2.8 to 3.5 for runs alone, a ratio of 4.7 to 6.2. So the rerun axis is roughly five times the smaller, and the cheaper to compute. Design effects agree: each configuration's 79 tasks at $k = 8$ carry about 97 to 111 independent observations, not 632. A paper running three seeds and printing a small error bar has measured the smaller component. On a suite this size no score is meaningful to better than about ten points, however often it is rerun. The ordering repeats on a reused corpus of 4 model families, 10.5 against 0.6 points.

Table 3: Both variance axes on every domain, in points, the two declared extensions included. Task axis resamples tasks and runs, rerun axis runs only. The D1 row is `claude-sonnet-5`; Table 2 gives all 6 configurations. Appendix A.8 adds design effects and unanimity counts.

Domain	Tasks	k	Runs	Task axis	Rerun axis	Ratio	Flip
VisualWebArena (D1)	79	8	3792	17.6	2.8	6.2	0.203
CRMArena-Pro B2B (D2)	119	8	952	15.0	3.4	4.5	0.395
τ-bench retail (D3)	115	8	920	13.7	4.3	3.2	0.617
CRMArena-Pro B2C	118	8	944	15.4	3.0	5.2	0.322
τ-bench airline	50	8	400	20.2	6.8	3.0	0.640

Two more domains, and the noise is concentrated. A benchmark checked a different way should not reproduce a harness artifact. Table 3 gives both axes on all five settings. τ-bench retail and CRMArena-Pro's frozen stratified split reproduce the direction under two unrelated checker designs, and τ-bench airline replicates retail on a disjoint task set. Four settings, three checker designs, one direction. P2 asked for flips above 10 percent on D1 and D3 and strictly lower on D2. The first two clauses hold overwhelmingly, the third fails because D2 flips at 0.395, above D1's 0.203 to 0.304, so P2 fails as a conjunction while the finding stands (Appendix A.12). That noise is also concentrated. Across the three Claude tiers only 17, 16 and 18 of 79 tasks are mixed, the rest unanimous failures (Appendix A.8). So four fifths of tasks are deterministic for a given agent, and the whole flip rate lives in the remaining fifth. That says where a rerun budget belongs, if those tasks can be found in advance, which Section 7 tests. It also explains the high settleability here, 0.595 to 0.797: tasks are settled because they are lost.

A live rank inversion. Two frontier models separated by 0.8 points invert under a joint task and run bootstrap with probability 0.404, at exact McNemar $p = 0.688$. Of 15 pairs, 3 sit within three points, every one inverting with probability 0.221 to 0.459. The starkest case is inside one family, 0.5 points inverting at 0.459, and the largest within-family spread 0.092 exceeds the spread between family means, 0.075 (Appendix A.9). P5 predicted a third of near-tied pairs would invert, but inversion is an event per resample rather than a property of a pair, so P5 fails as worded. Three diagnostics rule out the easier explanations. Aggregating replicates before testing hides most discriminating tasks, 19 to 44 percent retained (Appendix A.13), and container state explains nothing, 4 containers at $p = 0.946$ (Appendix A.14). Nor does temperature 0 remove them: greedy decoding, fixed seed, batch size one and eager execution still flip **0.133** of 30 tasks, so **a nonzero share of the web domain's nondeterminism is not policy sampling** (Appendix A.29). **That residue is the browser's, not agent evaluation's.** because a second domain did not reproduce it. τ-bench retail

carries a second stochastic actor: its header records temperature 0 and also a model playing the customer, whose opening utterance differs across all eight shipped trials. Pinning *both* the agent and the simulator to one greedy local endpoint removes the flipping entirely, 0 of 30 cells at pairwise disagreement 0.000, against 0.767 for the sampled arm on identical tasks, exact McNemar $p = 0.000$. So there every flip is policy sampling, registered as reportable either way (Appendix A.30). Reruns are the cheaper axis, but they are how a task gets settled, so how often does that work?

6 SETTLEABILITY MEASURES THE SCORE, NOT THE CHECKER

τ-bench settles 0.000 of its tasks and VisualWebArena settles 0.755. Read naively, τ-bench's checker must be worse. That is wrong: settleability tracks the spread of the per-task pass rates, not their average. VisualWebArena's agents pass 0.254 of runs, and 81.3 percent of its settled tasks are settled *failures*, so eight runs there mostly establish that a task is hopeless. The instructive pair is the other two, passing at effectively the same rate, 0.595 and 0.572, yet CRMArena-Pro settles 47 tasks against τ-bench's nothing. **The mechanism.** A shared beta prior fits each task's rate, and its concentration sets the direction of shrinkage: 0.488 on CRMArena-Pro pushes a task *outward*, 1.087 on τ-bench pulls it *inward*. So a perfect record holds 0.933 posterior mass above $1 - \alpha$ there, clearing the 0.9 required, against 0.831 on τ-bench, which clears nothing and so counts 0. Settling one such task needs $k = 12$ against $k = 6$ (Appendix A.11).

P1, and where it holds. Varying both stringency parameters over nine settings moves the estimates a lot: 0.000 to 0.755 on VisualWebArena, 0.000 to 0.605 on CRMArena-Pro, 0.000 to 0.383 on τ-bench (Table 11). The verdicts move less. Every setting falls below 0.70 on τ-bench and CRMArena-Pro, so P1 is supported on both regardless of convention. VisualWebArena's 0.755 is not supported and its range crosses the threshold, so that failure is convention dependent. Across 3 domains P1 holds on 2, so P1 is supported, and we report the split rather than averaging domains that disagree. A declared extension makes the mechanism concrete: the CRMArena-Pro design re-run over different organisation records moves settleability from 0.395 to 0.678 (Appendix A.12).

Pass rates near zero make a suite look settleable and middling rates do not, so as agents improve a fixed suite at fixed k becomes *less* able to settle its own tasks. Settleability belongs next to a score and cannot be inherited from prior work. Section 5 put the flipping in about a fifth of tasks, exactly the ones reruns cannot settle. Finding them in advance would target the budget, and a difficulty label is the obvious candidate.

7 DIFFICULTY IS NOT MEASURABLE AT FEASIBLE k

Table 4: Shipped difficulty labels against measured behaviour, 78 tasks, 6 configurations, $k = 8$, pooled over the 5 scoreable ones. Spearman is against label rank, so a working label comes out negative.

Label axis	Easy	Medium	Hard	Spearman	p
`overall`	0.415	0.172	0.239	-0.224	0.049
`reasoning`	0.328	0.195	0.177	-0.146	0.204
`visual`	0.285	0.245	0.272	+0.008	0.948

Stratifying by difficulty is the natural response, and two literatures move that way. Item response theory estimates item difficulty and selects informative items, and an efficiency argument evaluates only tasks whose historical pass rates sit near 0.3 to 0.7 (Appendix A.10). Both presume an item has a difficulty estimable from affordable data. VisualWebArena ships author-assigned labels on three axes, and Table 4 shows they do not support it. The strongest axis reaches a Spearman of only -0.224 and the `visual` axis is indistinguishable from nothing at +0.008. The levels are not even ordered: medium tasks pass at 0.172 against 0.239 for hard, monotonicity holding in 1 of 5 configurations. Difficulty is also only mostly a task property, 0.592 of run-level variance on the task main effect against 0.148 on the interaction. **Worst, the band selected for efficiency is the least measurable band.** Within-task variance is $p(1 - p)$, peaking at one half. In the 0.3 to 0.7 band, 10 tasks here, flips run 0.620 and settleability 0.000, against 0.082 and 0.918 in the extreme bands. Selecting informative items is not the error. Leaving unstated that it changes each task's rerun budget is.

8 REPAIR: WHAT TO BUY, AND WHEN TO STOP

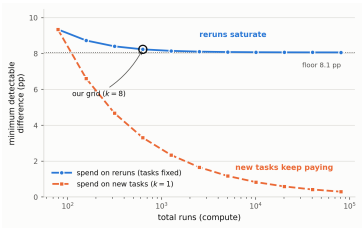


Figure 1: Measurement precision against compute, spent two ways, from the fitted two-level model. Lower is better, and both curves start from one pass over the suite. Reruns converge to a floor of 8.1 points set by how much tasks differ, while new tasks improve without bound. The circled point is our design.

So the flipping tasks cannot be found from a label. That leaves the allocation question in general form: given a fixed budget and no way to target it, how should it be spent? Under the two-level model, the variance of a score is

$$\mathrm{Var}(\widehat{\text{score}}) = \frac{\mathrm{Var}_{\text{between}}(p) + \mathbb{E}[p(1-p)]/k}{n_{\text{tasks}}}. \tag{1}$$

Tasks divide both terms of Equation 1. Reruns divide only the second. That asymmetry belongs to the estimator, so **the rerun axis saturates while the task axis does not**. At 79 tasks the minimum detectable difference is 9.3 points at $k = 1$, 8.2 at $k = 8$ and 8.1 at $k = 1024$, so a 128-fold increase past $k = 8$ buys one tenth of a point (Figure 1). Spent on new tasks, the same compute reaches 0.3 points, 27.6 times better. At a fixed budget of 632 runs the best allocation is $k = 1$ at 3.3 points and the worst $k = 16$ at 11.6, 3.5 times worse for identical spend. P6 is supported. This indicts our own design: we ran $k = 8$, inefficient for a score, because this paper needs the within- task distribution $k = 1$ cannot give.

A protocol that may stop early, and the price of its guarantee. Checking results as runs accumulate invalidates a fixed-n interval, while anytime-valid confidence sequences survive it (Howard et al., 2020; Waudby-Smith & Ramdas, 2024). We registered at least 0.95 agreement with the exhaustive grid at no more than 0.40 of its budget, and observed 0.533 at 1.00, so **P7 fails**. It is underpowered, not broken: at 2 times the suite size the same sequence does decide, so **an anytime-valid decision costs roughly 2× the tasks a fixed-n test needs** (Appendix A.31).

9 O2, JUDGEMENT: WHAT A PROGRAM CAN DECIDE, AND WHAT NEEDS AN OPINION

Everything so far treats the checker as given. Two things can go wrong with that. The check may not be decidable by any program, which is O2. Or the evaluator may never have discovered the capability it judges, which is O4. We measure both, on benchmark checkers and on a registry of agents deployed in the wild. The two populations answer oppositely, and that contrast is the finding. Two terms carry the section. A *check obligation* is one thing the checker must establish before calling a task passed, not one task: a checker requiring three particular strings carries three, and τ-bench's 115 tasks generate 644. Counting obligations stops a domain of one-obligation tasks dominating. An obligation is *contract decidable* if a program can decide it from the declaration alone, as string, DOM, URL and reference-action comparisons all can. What is left is *judgement residue*: was this answer the same thing as the reference, and was a refusal right.

Benchmark checkers are mostly programs. P3 asked whether at least 0.25 of check obligations across the three live domains are judgement residue. Counting obligations over each domain's full released set, the share is 0.065 on VisualWebArena, 0.000 on τ-bench and 0.206 on CRMArena-Pro, pooling to **0.140** over 3,446 obligations, so P3 is not supported. Equal domain weighting gives 0.090, and the residue sits in the few task families asking for a similarity judgement or a refusal (Appendix A.4). Its second clause asked whether judges are unreliable there. A panel of 7 annotators across three model families, three samples each, 840 calls over 40 residue obligations, reaches mean self consistency 0.988 against a registered "below 0.9" and cross-judge Krippendorff

alpha 0.761 against a registered "below 0.7". So this clause is not met, and both clauses fail the same way. Refusals are harder, 0.719 on 21 against 0.815 on 19 for similarity judgements. The parse rate was 0.929, unparsed calls excluded rather than counted as agreement.

Declarations by real agents are mostly judgement residue. For each of the 44 declared skills in a live registry's verified pool we asked whether a deterministic conformance check is derivable from the declaration alone. Seven annotators spanning three model families labelled every skill at five samples each, 1540 annotations at parse rate 0.998. Only 11 of 44 skills admit a deterministic check, a fraction of 0.250, Wilson interval $[0.146, 0.394]$. The remaining 33 require judgement, and across annotations a third flag the declaration as too vague to derive any check. P10 is supported, and it thresholds a fraction, so its verdict does not depend on the unpinned earlier read of 69 skills the registration named. Three quarters of what a curated, verified set of agents declares is not mechanically verifiable from what its authors wrote.

Worse, the boundary depends on the vendor consulted. Per-family decidable rates on the same 44 skills are 0.386, 0.182 and 0.273, between-family agreement 0.590 against within-family self consistency 0.578 to 0.947. So a judge is required to decide which obligations require a judge, and vendors disagree about that (Appendix A.24). The two panels side by side localise the obstruction. Cross-annotator and between-family alpha are 0.761 and 0.749 on benchmark residue, against 0.513 and 0.590 on registry declarations. **The judgement obstruction is not uniform across populations**, and residue moves with the difficulty of judging it.

The tempting reading is that it binds where an author wrote prose, not where a designer implemented a checker. A third population breaks it. The identical panel on VisualWebArena's residue gives 0.490 over 23 obligations, against 0.761 on CRMArena-Pro residue and 0.513 on registry declarations. Benchmark residue in one domain is judged *less* consistently than author-written prose, so who wrote the obligation is not the variable. **What the obligation is about is.** Split that domain by whether the task refers to something shown in an image: alpha is **0.357** on the 13 image-referencing obligations against 0.658 on the 10 others. A text panel settling questions about a picture it cannot see agrees barely better than chance, so an agreement number means nothing without saying what the obligations are about (Appendix A.25).

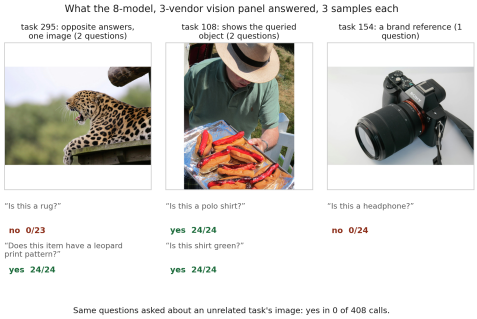


Figure 2: A checker we expected to fail, and did not. Each panel gives a real task image and what our 8-model panel answered at three samples. The left panel is the control that matters: it answers no to "is this a rug" and yes to "does this item have a leopard print pattern", which wording alone could not do.

One checker we expected to be invalid, and was not. Decidability is not validity, and our registered validity measurement returned a null. Here is one in its place, on a design the paper does not otherwise touch: an obligation settled by asking a vision model a closed question. 12 VisualWebArena tasks are settled this way and **all 17 stored answers are "yes"** (Figure 2). We predicted a broken checker and it is not broken: asked the same questions about an unrelated task's image, the 8-model, 3-vendor panel answered yes 0 times in 408, a gap of 0.759, every model between 0.706 and 0.780, none dissenting (Appendix A.26). So the obligation is decidable and the suite still cannot show it, because with no negative cases a constant-yes judge scores as highly. Reaching that number meant fixing two harness constants of ours, each defensible alone, together making a measurable clause unmeasurable (Appendix A.25).

10 O3, FAMILIES: WHICH FAMILIES YOU SEE DEPENDS ON WHO DREW THEM

Both O1 and O2 treat tasks as interchangeable items. If failures were independent across tasks, an aggregate score would summarise them and a random subset would estimate them. We tested that, embedding each task statement, grouping the embeddings with HDBSCAN (Campello et al., 2013), then asking whether group identity predicts failure under false discovery control. Between-group variance is real but far from universal. Across 6 model and corpus cells the ICC runs 0.000 to 0.434, and 2 cells show none, so coherent blind spots are a property of the agent, to be measured, not a law. Only 0 groups survive the correction: conditional pass rates vary a lot, some groups at zero against overall rates near a quarter, but individual groups are too small to survive multiplicity. P4 fails.

The null was about our grouping, not about the agent. That is testable, because two domains ship groupings we did not choose. VisualWebArena assigns every task an intent template, so tasks sharing one ask the same question about different products, and CRMArena-Pro ships task categories. On the *same runs* that gave 0 surviving groups under our own pattern matching, the template partition has per-configuration ICC 0.337 to 0.431, pooling to 0.366. And **12 of 17 templates** differ from the rest of the suite after the same correction, rates 0.000 to 0.758 against a pooled 0.260. CRMArena-Pro's categories give 12 of 17 at ICC 0.370. Two domains, two kinds of given grouping, one answer. This does not rescue P4, graded on the grouping we registered. It changes what the null means: we could not detect structure, rather than there being none. Templates are coarse another way, and 15 hold too few tasks to compare, so we count those rather than drop them. That defect also retired a registered threshold: a design effect shrinks as a grouping gets finer even at unchanged correlation, so we report ICC as primary (Appendix A.3).

11 O4, REACH: THE EVALUATOR CANNOT EVEN ENUMERATE THE POOL

Everything so far bounds what a verdict can say about evidence in hand. O4 asks the prior question: before a checker can be decisive, something has to enumerate the capabilities the evidence is meant to exercise. We audited the verified agent pool of a live registry: 18 agents at owner-specified pinned versions plus 5 local archives, 23 total, excluding unverified previews. Pinning is load bearing: the pinned versions differ from the registry's search index, and two pool agents are absent from it, so an audit reading the index measures a different artifact. Each agent has two possible declaration surfaces, a card served at the well known discovery path and a card packaged in the artifact. The split is exactly complementary: 0 agents expose both and 0 expose neither, so Table 17 has two populated rows. Discovery alone enumerates 7 of 18 agents, 39 percent, and the artifact alone 11, 61 percent. **Neither surface covers the pool**, so a checker on either assumption silently skips a third to two thirds of the agents it reports on.

The same result on a registry we did not build. We helped populate the pool above, so we repeated the measurement on the public Model Context Protocol registry, a third party. A resumable pull over 506 journalled pages returned 15,042 current servers, over 800 times the pool above. The index surface enumerates nothing, established from the union of every key across all 50,500 records: 12 keys, of which **0 name a tool, a capability or a skill**. On a frozen stratified probe of 40 servers given a real handshake, **22 answered, 0.550 with interval [0.398, 0.693]**, declaring 316 tools. Of the 18 that did not, 14 returned an authentication error, so the servers a live-surface checker skips are closed to it rather than broken, and 5,634 more publish no live endpoint (Appendix A.21). The obstruction belongs to registries. **A complete declaration can still be unusable.** 22 of 23 agents have a recoverable declaration, together declaring 44 skills, but completeness does not imply usability: **22 of 23 declared endpoints are loopback addresses and 0 are routable**, and protocol versions split 18 against 4 inside this one curated pool (Appendix A.27). Decidability is also separate from validity, which asks whether the reference is right; our validity measurement returned a null after gating, reported as attrition rather than a rate (Appendix A.23).

12 ATTRIBUTION: IS THE PRODUCT THE SCAFFOLD OR THE MODEL?

If task identity carries this much of the variance, how much is left for the thing a leaderboard compares? An agent product is a scaffold wrapped around a base model, and product scores cannot

Table 5: The nine confirmatory predictions of Table 9 under Holm-Bonferroni at family-wise $\alpha = 0.05$, as registered; 3 survive. P2's observed column is the flip rate on D1, D2 and D3, P8's the model share and the interaction. †fails on a non-test clause.

	observed	Holm p	verdict		observed	Holm p	verdict
P1	2 of 3 domains	$< 10^{-3}$	**supported**	**P6**	ratio 27.6	$< 10^{-3}$	**supported**
P2	0.245/0.395/0.617	$< 10^{-3}$	not supported†	**P7**	0.533 at cost 1.00	1.000	not supported
P3	0.140 residue	1.000	not supported	**P8**	0.902/0.076	1.000	not supported
P4	0 blind spots	1.000	not supported	**P10**	0.250 decidable	0.004	**supported**
P5	0/3 invert	1.000	not supported				

separate the two, because most pairs differ in both. Separating them needs a crossed design: 78 tasks $\times$ 6 models $\times$ 2 scaffolds $\times$ $k = 8$, balanced, 7,488 runs, zero harness failures. The two scaffolds differ by exactly one file, so this contrast bounds the scaffold effect from below. The 12 products span 13.9 points, where a leaderboard stops. Of the between-product variance, the base model accounts for 0.902 and the scaffold for 0.023. Directly, the largest model contrast moves the score 12.4 points, the one-file scaffold contrast 1.3 points downward, interval -2.5 to -0.1. That interval sits just below zero, so the scaffold effect is barely distinguishable from nothing next to a model effect ten times larger.

The interval is wide, and no rerun budget fixes it. The model share carries a bootstrap interval of 0.485 to 0.987, crossing the 60 percent threshold we registered, so that clause is unresolved rather than passed. The interaction clause fails at 0.076 against the registered 10 percent, so P8 is not supported. Six models and one scaffold contrast give few degrees of freedom, and adding runs shrinks the residual rather than the numerator, so the fix is more scaffolds (Appendix A.15). Note the denominator too: everything distinguishing these 12 products is 0.9 percent of the variance in whether a run passes, against 0.594 for task identity and 0.257 for within-task nondeterminism. The quantity leaderboards exist to compare is a small residue on two larger sources the score conceals. Nor is every attribution failure about a model: a hardcoded 2000 token output cap on a third benchmark turned correct reasoning into recorded failures, preferentially on the tasks needing most reasoning (Appendix A.22). Every obstruction so far makes a claim weaker. The last makes a false claim stronger.

13 O5, SHORTCUTS: WHY CHEATABILITY BELONGS TO THE INTERFACE

Recent audits find exploitability widespread. BenchJack reports working exploits across every suite it examined, τ-bench and WebArena included, on which our environment is built (Wang et al., 2026), and a complementary line embeds hacking opportunities so exploitation is verifiable by construction (Roth et al., 2026). Our contribution is not another audit. It is that these are statements about an interface as much as a benchmark.

Table 6: Exploit surface of our own 79-task VisualWebArena suite, given the reference answer file the harness ships. All three rows run against the harness's own evaluator, the last two against a live container. Controls score 0.000, 0.000 and 0.000, so the cheating comes from the leaked file, not a permissive checker (Appendix A.19).

Eval type	Tasks	What the file leaks	Status
`string_match`	33	the answer itself	demonstrated, 33/33 pass
`url_match`	16	the target URL	demonstrated, 14/16 pass
`program_html`	30	the target page state	demonstrated, 1/30 pass
Demonstrated without competence	48		0.608 of the suite

Our own harness ships the answers. Our vendored tree contains env/visualwebarena/config_files/vwa/test_shopping.raw.json, holding every task's reference answer in plaintext beside the runner, and all 79 of our tasks appear in it. We wrote a twenty line agent reading that file and returning the reference answer, taking no browser actions and making no model calls. Scored by the harness's own evaluator it reaches 1.000 on the 33 string-match tasks, against controls of 0.000 and 0.000. Table 6 gives all three arms, Appendix A.19 their mechanics. The `program_html` arm resists, because for all but 4 specs the evaluator re-navigates and reads server state, wiping whatever the agent did. So **48 of 79 tasks, 0.608, are passable with no task competence**, and cheatability tracks what the verdict reads: a checker trusting the returned string

falls completely, one trusting the final URL mostly, one re-reading server state resists. The best genuine configuration scores 0.261 against the file-reading agent's 0.418, and a leaderboard cannot tell them apart.

A second suite, and a checker hiding the answers cannot repair. If cheatability tracks what the verdict reads, it should transfer to an unrelated checker. CRMArena-Pro's evaluator short-circuits at line 527 before any model call: when the reference is single valued, a returned string equal to it scores one. On the frozen 119-task split an agent reading the dataset and echoing the reference scores **113 of 119, 0.950**, against a genuine 0.588, with 0 model calls and 0 org calls. Controls hold at 0.000, and the degenerate control our web arms lack scores 0.185: 22 references are null and the checker rewrites them to the string "None", so the exploit reads as 113 against 22 rather than zero. 16 of 17 categories are fully cheatable (Appendix A.20). The consequence is sharper than for a leaked file: **hiding the dataset would not repair this checker**, because the short-circuit compares against a value the benchmark needs to grade.

None of this was reachable in the runs of Section 5, by design. Our scaffold exposes five browser primitives and no file read, no shell and no arbitrary navigation, so the answer key sits one directory from the agent's process, closed to it (Appendix A.19). One benchmark therefore carries two integrity properties: inert under a browser-only interface, decisive under one with a shell. **The action space is the load-bearing variable**, so a certificate must name the one it was issued under. And the field is removing this protection: general computer use is the direction of travel, and every added capability enlarges the set of verdict-determining objects the agent reaches. Benchmark integrity has rested on interface poverty.

Table 7: One remedy per obstruction, as a procedure rather than a reporting item, with the measurement of ours that motivates it and what it costs. Appendix A.17 gives all five in full, the cases where one cannot be followed included.

	Procedure	Motivated by	Cost
O1	Spend the budget on tasks, not on reruns	detects 3.3 points at $k = 1$ against 11.6 at $k = 16$, same 632 runs	per-task rates become noisy, so tasks cannot be ranked
O2	Return undecided on the residue; span three or more vendors	residue 0.140 on benchmark checkers, 0.750 on declarations; between-vendor agreement 0.590	a panel three times the size, and a lower rate than one vendor reports
O3	Prefer a grouping the benchmark authors supplied; report the correlation, not the design effect	12 of 17 templates and 12 of 17 categories survive, on runs where our own grouping found none	such a grouping may not exist, and then the claim is about the analyst
O4	Read every declaration surface, pin versions, count what none enumerates	7 agents declare only at discovery, 11 only in the artifact; the MCP index enumerates 0 tools for 15,042 servers	coverage numbers get worse, and some pools admit no complete enumeration
O5	Make the verdict read state the agent cannot write	re-reading server state resists at 1 of 30; trusting a returned string falls at 0.950	the evaluator needs privileged environment access, which is why cheap checkers dominate

14 THE INSTRUMENT AUDITS ITSELF, AND WHAT A REPORT SHOULD CARRY

This paper argues that agent evaluation instruments are unaudited, so we turn the same scepticism inward. Six quotable findings of ours dissolved on inspection, each caused by our own instrument: output truncation faked a compliance gap, a single-family panel agreed at 0.739 against 0.513 across families, completion-order listing faked an alternating flip pattern, a text-only panel judging image grounded instructions faked 85 percent invalid checkers, a zero-step adapter defect faked two weak families, and an eleven point gap was 1.6 points on one subset (Appendix A.28). The registered family is corrected the same way. Of nine predictions under Holm, 3 survive (Table 5), Holm reordering nothing. Two readings changed under strict grading, which is the point of registering them, and both are one defect: a threshold on a quantity that depends on how it is read (Appendix A.2).

A reporting standard, and what this study cannot settle. Each obstruction implies one cheap reporting item, and where one binds, the output should be undecided with the obstruction named. Appendix A.16 states those five, Appendix A.18 our limits. Reporting is not repair, so Table 7 gives one remedy per obstruction as a procedure, with its motivating measurement and its cost (Appendix A.17). None needs a new method, and two are worth naming. The rerun axis is the wrong place to spend a budget. And secrecy repairs nothing, since CRMArena-Pro compares against a value the benchmark must contain to grade. What has to change is what the verdict reads.

REFERENCES

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57(1):289–300, 1995. doi: 10.1111/j.2517-6161.1995.tb02031.x.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, et al. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint*, 2024.

Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 160–172. Springer, 2013. doi: 10.1007/978-3-642-37456-2_14.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent llm systems fail? *arXiv preprint*, 2025.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1): 1–26, 1979. doi: 10.1214/aos/1176344552.

Chris Ge, Daria Kryvosheieva, Daniel Fried, Uzay Girit, and Kaivalya Hariharan. Agent psychometrics: Task-level performance prediction in agentic coding benchmarks. *arXiv preprint*, 2026.

Steven R. Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform chernoff bounds via nonnegative supermartingales. *Probability Surveys*, 17, 2020. doi: 10.1214/18-PS321.

Han Jiang, Sunbeom Kwon, Jinwen Luo, Ziang Xiao, and Susu Zhang. Can we trust item response theory for ai evaluation? *arXiv preprint*, 2026.

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

Leslie Kish. *Survey Sampling*. John Wiley and Sons, 1965.

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*, 2024.

Klaus Krippendorff. *Content Analysis: An Introduction to Its Methodology*. SAGE Publications, 4th edition, 2018.

Peiyu Li, Xiuxiu Tang, Si Chen, Ying Cheng, Ronald Metoyer, Ting Hua, and Nitesh V. Chawla. Adaptive testing for llm evaluation: A psychometric alternative to static benchmarks. *arXiv preprint*, 2025.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, et al. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: A benchmark for general ai assistants. *arXiv preprint*, 2023.

Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations. *arXiv preprint*, 2024.

Franck Ndzomga. Efficient benchmarking of ai agents. *arXiv preprint*, 2026.

Anka Reuel, Amelia Hardy, Chandler Smith, Max Lamparth, Malcolm Hardy, and Mykel J. Kochenderfer. Betterbench: Assessing ai benchmarks, uncovering issues, and establishing best practices. In *Advances in Neural Information Processing Systems*, 2024.

Amit Roth, Ankur Samanta, Matan Halevy, Yoav Levine, and Yonathan Efroni. Hack-verifiable environments: Towards evaluating reward hacking at scale. *arXiv preprint*, 2026.

Hao Wang, Hanchen Li, Qiuyang Mang, Alvin Cheung, Koushik Sen, and Dawn Song. Do androids dream of breaking the game? systematically auditing ai agent benchmarks with benchjack. *arXiv preprint*, 2026.

Ian Waudby-Smith and Aaditya Ramdas. Estimating means of bounded random variables by betting. *Journal of the Royal Statistical Society: Series B*, 86(1):1–27, 2024. doi: 10.1093/jrsssb/qkad009.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953.

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

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

Hongli Zhou, Hui Huang, Ziqing Zhao, Lvyuan Han, Huicheng Wang, Kehai Chen, Muyun Yang, Wei Bao, et al. Lost in benchmarks? rethinking large language model benchmarking with item response theory. *arXiv preprint*, 2025.

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

Table 8: Every term this paper uses. Section 3 states the two that carry the argument; each later section is an application of one of these.

Term	What it means
Task axis	How much the score would move had we drawn a different sample of tasks from the same population, holding reruns fixed.
Rerun axis	How much it would move if we reran the same tasks, holding the task set fixed. This is what a seed-to-seed error bar reports.
Flip	A task *flips* when the same agent, rerun on it under the same conditions, both passes and fails it. Nothing about the task changed, so the verdict is a property of the draw. A task passing all k times, or failing all k times, does not flip.
Flip rate	The share of a suite's tasks that flip: the fraction on which a single run would have told you the wrong thing, depending on the draw. We hold $k = 8$ when comparing, since a deeper budget finds more flips on the same suite. Where depth differs we report pairwise disagreement, which does not grow with k.
Settled task	A task whose reruns pin its success probability to one end, so a further run adds nothing. Formally, at least $\gamma = 0.9$ of its posterior lies outside $[\alpha, 1-\alpha]$, at $\alpha = 0.1$.
Settleability	The share of a suite's tasks that are settled. Called the *decisive fraction* elsewhere in the literature and in our code.
ICC	The share of run-to-run variation explained by which task the run was on, rather than by nondeterminism within a task.
Design effect	How many runs it takes to learn what one independent observation would. A design effect of 6 means 3792 runs carry a sixth of that many independent observations.
Check obligation	One thing a checker must establish before calling a task passed. Not one task: a task needing three strings on the page carries three obligations.
Judgement residue	The obligations no program can decide, so a judge is required. Similarity and refusal judgements are residue; string, DOM, URL and tool-call comparisons are not.
Action space	The primitives the agent can actually execute. Our web agent has five: click, type, press a key, scroll and finish.

A Appendix

Nothing in this appendix is a new result. Each subsection is material moved out of the body, retained in full so that no measurement in this study is lost. The body states the primary result for each obstruction, and every body section names the subsection here that expands it. This appendix carries the derivations, the secondary measurements and the discussion.

A.1 The vocabulary, and the estimator definitions

Table 8 states every term in plain words. We report per-task success probabilities p_t from a beta binomial hierarchical fit rather than raw fractions.

The *decisive fraction*, which the body calls settleability, is the share of tasks whose posterior places at least γ mass outside $[\alpha, 1-\alpha]$, with $\gamma = 0.9$ throughout. It answers directly whether one further trace could settle a verdict. Thresholding a posterior *mean* instead is wrong at small k, because the mean is then dominated by the prior; Section 14 reports the false result that produced.

The *two axis interval* combines a cluster bootstrap over tasks (Efron, 1979) with binomial resampling of runs within task. We report three intervals: task only, run only, and both. Published scores typically report neither of the first two, and the gap between the third and the second is the central quantity of this paper.

The *design effect* is $1 + (\bar{k} - 1)\,\mathrm{ICC}$ (Kish, 1965), with effective sample size $n_{\text{eff}} = nk/(1 + (k - 1)\,\mathrm{ICC})$. A design effect quoted without its mean group size is not interpretable, for the reason Section 10 gives.

Table 9: The nine registered predictions in plain words, with the section that tests each. The thresholds are as registered, not as chosen afterwards. Verdicts under the family-wise correction are in Table 5.

	Name	What we predicted, before the data	Tested in
P1	settleability	On at least 2 of 3 domains, fewer than 0.70 of tasks are settled by eight reruns.	§6
P2	flipping	Identical reruns flip the verdict on at least 10 percent of tasks on D1 and D3, and strictly less often on D2.	§5
P3	judgement share	At least 0.25 of what benchmark checkers must establish needs a judge, and judges are unreliable on that part.	§9
P4	families	Failure is not independent across tasks: at least two blind-spot groups per agent survive multiplicity control.	§10
P5	rank stability	At least a third of configuration pairs within 3 points invert when the runs are resampled.	§5
P6	allocation	Above $k = 3$, adding tasks buys at least twice the ranking certainty per unit compute that adding reruns does.	§8
P7	stopping rule	A protocol that may stop early agrees with the exhaustive grid at least 0.95 of the time, at no more than 0.40 of its cost.	§8
P8	attribution	The base model explains at least 60 percent of between-product score variance, and the scaffold by model interaction at least 10 percent.	§12
P10	skill decidability	Fewer than half of the skills that real agents declare admit a program check derived from the declaration alone.	§9

A.2 THE PRE-REGISTRATION, AND THE LIMIT OF ITS EVIDENCE

Section 4 claims the thresholds are ex ante and offers the repository history as the check. This states what that history can and cannot establish.

It establishes order, not dates. A root commit has no earlier commit to anchor its timestamp, and a timestamp is in any case writable. So we claim only what preceded what: the registration commit is the root, it contains no trajectory data and no draft, and all 12 result files behind a confirmatory prediction were first committed after it. The annotated tag is named after the review period rather than after us, and releasing it at camera-ready lets a reader verify that ordering independently.

The append-only property is the load-bearing one. 561 lines have been added to the file and 0 removed across its whole history. A moved goalpost requires editing a threshold, which is a deletion, so the deletion count is the thing to audit rather than the prose. Our number generator fails the build if that count is ever nonzero, and fails it again if the registration commit stops being the repository root. Departures from the registered plan are recorded in a separate deviations log, which is append-only for the same reason and is not permitted to touch a threshold.

Two registered readings changed under strict grading, and Section 14 counts both as one defect. P2 requires flips above 0.10 on two domains *and* a strictly lower rate on the third, so its third clause fails on the numbers in Section 5. P5 counts pairs that "invert", but inversion is an event per resample rather than a property of a pair, so no close pair qualifies however often it flips, which is the reading Section 5 applies. In both cases the threshold was placed on a quantity whose value depends on how it is read, and registering them is what exposed that.

Two of the ten registered items are not confirmatory tests of ours, and we separate them for opposite reasons. P9's sweep was committed in the freeze commit itself, so it is a measurement rather than a prediction, and Section 4 gives the argument. P10 is the contrast that shows the rule is not applied selectively: it draws on the same corpus, but it asks a question that was open at the freeze and its result file postdates the freeze, so it stays in the family. The design-effect threshold was retired during the study, for the reason Appendix A.3 gives, and it is recorded as retired rather than quietly dropped.

A.3 P4 ON THE BENCHMARK'S OWN GROUPING

Section 10 reports that VisualWebArena's own intent-template grouping finds structure our registered pattern matching did not. Table 10 gives that test template by template, which the body summarises in one sentence.

The grouping defect also retired a registered threshold, which Section 10 states in one clause. Design effect is $1 + (\bar{k} - 1)\,\mathrm{ICC}$, so it falls as the grouping gets finer even when the underlying correlation is unchanged. The same data met our registered design-effect threshold under a coarse grouping and failed it under a fine one. So we report ICC as the primary quantity, and quote design effects only alongside the mean group size behind them.

Table 10: Per-template pass rates on the VisualWebArena intent-template grouping, pooled over the scoreable configurations and tested against the rest of the suite with Benjamini-Hochberg control at $q = 0.1$ (Benjamini & Hochberg, 1995). Only templates with at least 3 tasks are tested, which is 17 of 32 templates. The 15 smaller templates still contribute to the ICC. 12 templates survive the correction.

template	tasks	pass rate	p vs rest	BH adjusted	survives FDR
64	3	0.000	0.0000	0.0000	**yes**
99	3	0.000	0.0000	0.0000	**yes**
101	3	0.000	0.0000	0.0000	**yes**
65	3	0.033	0.0000	0.0000	**yes**
91	4	0.075	0.0000	0.0000	**yes**
100	4	0.113	0.0000	0.0000	**yes**
76	3	0.167	0.0190	0.0294	**yes**
49	3	0.200	0.1378	0.1673	no
75	3	0.225	0.3980	0.4229	no
97	3	0.283	0.5264	0.5264	no
102	3	0.300	0.3391	0.3844	no
33	3	0.325	0.1109	0.1450	no
61	4	0.331	0.0414	0.0586	**yes**
69	3	0.367	0.0080	0.0135	**yes**
63	3	0.600	0.0000	0.0000	**yes**
68	4	0.613	0.0000	0.0000	**yes**
62	3	0.758	0.0000	0.0000	**yes**

A.4 P3: CLASSIFYING CHECK OBLIGATIONS

An obligation is one thing the checker must establish, not one task, and we take each domain's full released set rather than the subset we ran, so the share does not depend on our sampling. The classification is mechanical, was fixed before counting, and is deliberately generous to decidability. String, DOM and URL comparisons of any kind, reference action sequences compared by name and arguments, required output substrings, and CRMArena's `exact_match` all count as contract decidable. `fuzzy_match`, which the VisualWebArena harness itself defers to a judge, `page_image_query`, and CRMArena's `privacy_rejection` count as residue.

Per domain: 0.065 of VisualWebArena's obligations are residue, 0.000 of τ-bench's, and 0.206 of CRMArena-Pro's, pooling to 0.140 over 3,446 obligations, which is 483 residue obligations. One case is arguable and we flag rather than bury it. CRMArena's `exact_match` compares strings deterministically first and falls back to an LLM *extractor* when the agent's prose does not match verbatim. We count it decidable, because the obligation is decidable and the extractor only handles presentation. Counting those as residue instead gives 0.633, which would still leave P3 unsupported, so the verdict does not turn on the choice.

A.5 RELATED WORK IN FULL

Uncertainty and reporting. That benchmark scores carry sampling error is established. Miller (2024) derives error bars and argues for paired designs. Biderman et al. (2024) documents reproducibility failures from unreported harness detail. HELM (Liang et al., 2023) standardises reporting, and Reuel et al. (2024) scores benchmarks against a design checklist. These treat a benchmark as independent items scored by a fixed oracle. Agent evaluation breaks both assumptions, because the environment carries state and the oracle is itself a program or a model. Checklists also prescribe rather than measure. Our contribution is that the noise decomposes, that the component the field prices is the smaller one, and that two quantities a checklist asks for are not properties of a benchmark at all. The design effect depends on the analyst's grouping, and the share of obligations needing judgement depends on which model family is asked.

Agent benchmarks. We use the shopping environment of VisualWebArena (Koh et al., 2024), built on WebArena (Zhou et al., 2024), plus τ-bench (Yao et al., 2024), which adds a simulated user and introduces pass^k. Related suites include SWE-bench (Jimenez et al., 2024), whose hidden-test oracle is the closest common case to a deterministic checker, and GAIA (Mialon et al., 2023). pass^k summarises the same per-task Bernoulli structure we fit, but as a scalar. We keep the distribution, because it identifies which tasks carry the variance. Cemri et al. (2025) taxonomises multi-agent

failures by process, where we measure outcomes, because a process label is itself a judgement subject to the limits of Section 9.

Model-as-judge reliability. Zheng et al. (2023) establishes that model judges track human preference and documents their biases, asking whether a judge agrees with humans on a fixed task. We ask a prior question: for which obligations is a judge needed at all. That answer is itself judge-dependent, with between-family agreement of 0.590, so a single-family panel reports its own boundary as the task's.

Statistical machinery. Our estimators are standard and we claim no novelty in them: a cluster bootstrap (Efron, 1979) over tasks with binomial resampling within task, the survey-sampling design effect (Kish, 1965), exact McNemar (McNemar, 1947), Wilson intervals (Wilson, 1927), Benjamini-Hochberg (Benjamini & Hochberg, 1995), Krippendorff's alpha (Krippendorff, 2018), HDBSCAN (Campello et al., 2013), and anytime-valid confidence sequences (Howard et al., 2020; Waudby-Smith & Ramdas, 2024). The contribution is which one belongs at which stage, and what changes when the wrong one is used.

A.6 Setup in full

Our main grid is the shopping environment of a web agent benchmark, restricted to the 79 tasks our harness supports end to end. That is the entire eligible population, so no sampling rule is needed, and verdicts come from the benchmark's own programmatic checkers. 6 configurations across three families drive an identical scaffold at $k = 8$, giving 3792 runs, all pinned and logged per call. The two non-native families reach the scaffold through an adapter that translates the tool-use protocol, so the scaffold is byte identical across families and a family effect cannot be a scaffold effect in disguise. A self-hosted open weights configuration runs in two further arms. Runs execute against independent environment instances, one per worker, never shared concurrently, and we verify that instance identity does not predict the outcome. The scaffold cannot be pinned by commit, because the files that matter are untracked in its repository, so we content address it: every source file is hashed and the digest recorded per run. Every model call is journalled with its full request and response, and every numeric claim here is generated from a result file by a script.

A.7 Reproducibility

We release the estimator library with unit tests, the declaration corpus with both surfaces per agent, the sampling utilities that produce every task subset with a committed manifest and hash, and the reporting standard of Section 14. Predictions and analysis code were committed and tagged before the grids ran, and departures are recorded in a separate log rather than edited in place. Every number here is emitted by a script from a result file.

A.8 The two-axis decomposition on every domain

Table 3 in Section 5 gives both axis components for every domain, including the two extensions that no registered prediction counts. Two further quantities per domain belong with it, because they say how much independent evidence the runs carry.

On τ-bench retail the design effect is 4.35, so 920 runs carry the information of 211.3 independent observations, its ICC is 0.479, above the web suite's, and 44 of 115 tasks are unanimous. On CRMArena-Pro B2B, 72 of 119 are unanimous. Unanimity is the complement of the flip rate at fixed k, and it is the quantity Section 6 turns into settleability.

Section 5 states that the flipping concentrates in about a fifth of tasks, and that the unanimous rest are overwhelmingly unanimous failures. The counts behind that, on the three Claude tiers at 79 tasks, are 56, 52 and 48 always failing, against 6, 11 and 13 always passing. The asymmetry is what makes settleability high on this suite: a task settled at 79 tasks and $k = 8$ is usually settled as hopeless rather than as solved.

A.9 THE RANK INVERSION IN FULL

`claude-opus-5` scores 0.253 and `claude-sonnet-5` scores 0.261, a gap of 0.8 points with the nominally stronger model behind. Under a joint task and run bootstrap that ordering inverts with probability 0.404, and the interval on the difference runs from -6.6 to 4.7 points. Exact McNemar (McNemar, 1947) on paired per-task outcomes gives $p = 0.688$ on 6 discordant tasks.

There is therefore no evidence of a real difference in either direction, and a leaderboard reporting this ordering would be reporting a coin flip. We emphasise that these are current frontier models on a standard suite, not a constructed example.

Across all 15 pairs, 3 are separated by 3 points or less, and *every one of them* inverts with substantial probability, from 0.221 to 0.459.

The clearest case is within a single family. `gemini-3.7-flash` scores 0.305 against 0.301 for `gemini-3.1-pro-preview`, a gap of 0.5 points, with inversion probability 0.459 and McNemar $p = 1.000$. A vendor's cheap tier and its frontier tier are indistinguishable on this suite, and which one leads is a coin flip. That is a harder result to dismiss than a cross-vendor tie, because the two configurations share a tokenizer, an API and a training lineage.

A related observation bounds how much of a score is attributable to model family. The largest *within* family spread we observe, 0.092, exceeds the spread between family means, 0.075. Choosing a tier within a vendor moves the score more than choosing a vendor.

A.10 DIFFICULTY LABELS: FULL DISCUSSION

A natural response to a single benchmark number is to stratify by difficulty. Item response theory has been imported into language model evaluation to estimate item difficulty and ability and to select informative items (Zhou et al., 2025; Li et al., 2025), recently including agentic coding with an explicit split of ability into model and scaffold components (Ge et al., 2026). A parallel efficiency argument evaluates new agents only on tasks with intermediate historical pass rates, near 0.3 to 0.7, cutting task counts while preserving rankings (Ndzomga, 2026). Whether the estimation tools transfer to this data regime is itself questioned (Jiang et al., 2026). All of it presumes an item has a difficulty estimable from affordable data. VisualWebArena ships author-assigned labels on three axes, so we can test that.

The labels barely order behaviour. Pooled pass rates, easy to hard, are 0.415, 0.172, 0.239 on `overall`, 0.328, 0.195, 0.177 on `reasoning`, and 0.285, 0.245, 0.272 on `visual`. Spearman against label rank is -0.224 ($p = 0.049$), -0.146 ($p = 0.204$) and +0.008 ($p = 0.948$), so `visual` carries nothing (Table 4). The levels are also not ordered: medium tasks pass at 0.172 against 0.239 for hard, and the monotone ordering holds in 1 of 5 configurations. A stratified table on these labels would report levels that are not ordered for most systems.

Difficulty is mostly, but not only, a task property. A crossed decomposition puts 0.592 of run-level variance on the task main effect and 0.148 on the task by configuration interaction, with per-task rates correlating 0.760 between configurations. So a label estimated on one family transfers only that far.

Efficient selection buys the least measurable tasks. Within-task variance is $p(1-p)$, maximal at one half, so the intermediate band those proposals select for is where a task's own rate is least identifiable. In the 0.3 to 0.7 band, 10 tasks here, the flip rate is 0.620 and settleability is 0.000; in the extreme bands they are 0.082 and 0.918. This is not an argument against selecting informative items. It is that the selection changes the rerun budget each task needs, and that budget is left unstated. An adaptive protocol taking maximally informative items at one run each buys task-axis precision and pays on the rerun axis, which Section 8 prices. The labels give no guidance either, since flip rates across the three levels are flat.

Table 11: Settleability under nine stringency settings. A task is settled when its posterior puts at least *certainty* mass within *alpha* of either extreme. The registered setting is $\alpha = 0.1$, certainty 0.9, and P1 is supported on a domain whose value falls below 0.70.

α	Certainty	D1, VWA	D2, CRMArena-Pro	D3, τ-bench
0.05	0.8	0.755	0.395	0.000
0.05	0.9	0.614	0.000	0.000
0.05	0.95	0.000	0.000	0.000
0.1	0.8	0.755	0.605	0.243
0.1	**0.9**	**0.755**	**0.395**	**0.000**
0.1	0.95	0.614	0.000	0.000
0.2	0.8	0.755	0.605	0.383
0.2	0.9	0.755	0.605	0.383
0.2	0.95	0.755	0.605	0.243

Table 12: Fitted Beta priors per domain, and what they imply for settling a task. The shape column is the sign of $\hat{\alpha} + \hat{\beta} - 1$: a concentration below one shrinks a task outward, above one shrinks it inward. The last column is the posterior mass above $1 - \alpha$ for a task that passed at every one of that domain's own k. Domains with similar means settle very differently, which is the point.

Domain	mean	$\hat{\alpha}$	$\hat{\beta}$	shape	settled pass	settled fail	perfect-k mass
VisualWebArena	0.254	0.087	0.256	U-shaped	67	291	0.907
CRMArena-Pro B2B	0.595	0.290	0.197	U-shaped	47	0	0.933
τ-bench retail	0.572	0.622	0.465	interior-peaked	0	0	0.831

A.11 SETTLEABILITY: THE FITTED PRIORS AND THE SPLIT PER DOMAIN

Section 6 argues that settleability tracks the spread of the per-task pass rates rather than their mean. Table 12 gives the fitted parameters behind that argument for all three domains, including VisualWebArena, which the body discusses only in words.

Concentrations are 0.344, 0.488 and 1.087, so the U-shaped and interior-peaked cases both occur in our data. The perfect-record column is read at 8, 8 and 8 runs respectively, those being each domain's own k. Only two of the three clear the 0.9 bar.

The practical form of the same point is the minimum k at which a perfect record settles: 8 on VisualWebArena, 6 on CRMArena-Pro and 12 on τ-bench. We report the exact minimum over every k rather than the smallest entry of a display grid, since a coarse grid rounds the requirement upward and overstates the rerun cost.

Two things follow that the body states only for the pair it contrasts. VisualWebArena's settled tasks are 81.3 percent settled *failures*, against 0.0 on CRMArena-Pro and n/a on τ-bench, so what eight runs establish there is mostly that a task is hopeless. And the per-domain P1 verdicts are not supported, supported and supported at the registered setting. Support is threshold-independent on tau-bench retail and CRMArena-Pro B2B, meaning the whole stringency grid of Table 11 falls on one side of 0.70 there.

A.12 THE TWO DECLARED ROBUSTNESS EXTENSIONS

Both were declared as extensions before running, the B2C arm in its frozen manifest, so neither is a post-hoc reclassification and neither is counted toward P1's 2-of-3 condition.

τ-bench airline. 50 tasks at $k = 8$, 400 runs, score 0.445. Task axis 20.2 points against a rerun axis of 6.8, ratio 3.0. Flip rate 0.640 with 18 of 50 tasks unanimous. Settleability 0.000. This is the same picture as retail on a disjoint task set with a different policy document.

CRMArena-Pro B2C. 118 tasks at $k = 8$, 944 runs, score 0.640, task axis 5.2 times the rerun axis, flip rate 0.322. Settleability 0.678, interval [0.589, 0.756], against 0.395 on the primary arm. The settled split explains the 28 point move that Section 6 reports: the primary arm has 47 confident passes and 0 confident failures, the extension 57 and 23. The templates, the allocation and the seed are held fixed, so only the organisation records differ. 1 task is excluded, having failed 13 of 14 attempts identically with empty model content after roughly three hundred second trajectories. That is a reproducible harness limit rather than sampling attrition, and every attempt is retained in the journal.

A.13 AGGREGATION DISCARDS EVIDENCE

Two valid analyses of the same data disagree sharply. `gemini-3.1-pro-preview` leads `claude-sonnet-5` by 4.0 points. The joint bootstrap puts the inversion probability at 0.065, so it is fairly confident in the ordering. Exact McNemar on the same pair gives $p = 0.727$, which is no evidence at all.

The discrepancy is not a contradiction. It is a power difference, and its size is worth stating. McNemar needs one binary outcome per task, so the k runs must first collapse to a verdict. Any task where the two models differ in pass rate but agree in majority verdict then contributes nothing. For this pair, 26 tasks differ in rate and only 8 differ in majority verdict, so 18 discriminating tasks are invisible to the test. Across all 15 pairs, majority aggregation retains only 19 to 44 percent of the tasks on which the models actually differ.

The recommendation is concrete. Do not aggregate replicates to a verdict before testing. A task where one model passes 2 of 8 and another 0 of 8 carries real information, and collapsing both to *fail* throws it away. Test on the rates.

A.14 ENVIRONMENT STATE AND CONTAINER DRIFT

Because runs share a pool of environment instances, instance state is a candidate explanation for the flips. We declared this diagnostic before collecting the data. Across 4 instances a chi square test of pass rate homogeneity gives $\chi^2 = 0.37$ with $p = 0.946$, so there is no instance effect to speak of.

The finer evidence is more decisive than that aggregate. Of 7 flipping task cells, 5 had a *single* instance return both a pass and a fail. No account based on instance state can produce that. A between-instance comparison alone could not have separated the two explanations.

A.15 THE POWER BEHIND THE ATTRIBUTION NULL

P8 in Section 12 returns an unresolved model share and a failed interaction clause, so the question is what the design could have detected. A power study run before we saw the data excludes an interaction in which a scaffold helps a weak model eight points more than it helps a strong one. It does not exclude a four point version of the same effect. So the null is informative against a large interaction and uninformative against a moderate one, and the binding constraint is the number of scaffolds rather than the number of runs: adding reruns shrinks the residual term and leaves the between-product numerator where it was.

A.16 THE REPORTING STANDARD, ITEM BY ITEM

Section 14 says each obstruction implies one cheap reporting item. The five items are both variance axes, settleability beside the score, the residue share with the panel behind it, the grouping behind any structure claim, and the action space the verdict was issued under. This gives the reason for each.

Both variance axes, since one error bar cannot say which axis it summarises, and on every grid we ran the rerun axis was the smaller of the two, by 3.0 to 6.2 times. A seed-only error bar therefore reports the smaller component and reads as the whole.

Settleability beside the score, since it is a fact about this year's pass rates rather than about the checker, and Section 6 shows it cannot be inherited from prior work on the same suite. As pass rates leave the floor it falls.

The residue share and the composition of the panel behind it, since Section 9 shows the decidability boundary moves with the vendor consulted. A single-family panel inherits that family's boundary without recording it.

The grouping behind any structure claim, preferring one the benchmark authors supplied, since Section 10 shows a null can belong to the analyst's grouping rather than to the agent.

And the action space the verdict was issued under, since Section 13 shows the same suite is inert under browser-only primitives and 0.608 cheatable once a file read is reachable.

A.17 FIVE REMEDIES, AND WHAT EACH ONE COSTS

Appendix A.16 says what a report should carry. That is a reporting register, and a reader can comply with it and still run the same flawed study. This states each remedy as a procedure instead, with the number of ours that motivates it and the price of following it. One per obstruction. None of them requires a new method, and that is deliberate: every one is already paid for by evidence in this paper.

O1. Spend the budget on tasks, not on reruns. Section 8 measures this rather than asserting it. At a fixed 632 runs, allocating to 1 rerun per task gives a minimum detectable effect of 3.3 points, and allocating to 16 reruns gives 11.6 points, a factor of 3.5. The rerun axis saturates because task-to-task variance dominates, with an intraclass correlation of 0.744. The cost is that per-task pass rates become noisy, so a study that needs to rank individual tasks cannot take this trade. A study that needs to rank systems should.

O2. Return undecided on the residue, and span at least three vendors. Two parts, each with its own number. First, a certificate that reaches an obligation no program can settle returns undecided with the obligation named, not a score. This is not a rare case. The residue share is 0.140 on benchmark checkers and 0.750 on declarations written by agent authors, so the practice matters most exactly where evaluation is moving. Second, any judge panel standing in for a program spans three or more vendors. On registry declarations, agreement within a family runs from 0.578 upward while agreement between families is 0.590, so a single-family panel inherits that family's decidability boundary and records nothing about it. The cost is a panel three times the size of a single-vendor one, and a rate that is lower than a single-family panel would have reported. The gap between the two is the part that was never real. Two things this remedy deliberately does not do. It sets no threshold on agreement: across three populations measured through one code path, cross-judge alpha runs 0.761, 0.490 and 0.513 (Appendix A.25), so a fixed pass mark would be a statement about which population was sampled. Report the alpha with the population named instead. And it counts *vendors*, not model families. We wrote it the other way first, and trying to follow it on image obligations is what exposed the difference. Our text-only judge cannot take an image, so the panel had two vendors; the first extra vision model we found was from a vendor already in the panel, which satisfies the words and buys none of the training independence the rule exists for. The panel that finally satisfies S2 spans 3 vendors and 8 models, and assembling it needed one more correction of ours: a sweep concluded a third vendor was unavailable because it asked the global endpoint for a model published in one region only, where a missing model and a missing permission return the same status code.

O3. Prefer a grouping the benchmark authors supplied, and report the intraclass correlation rather than the design effect. Section 10 shows why this is not a stylistic preference. On the same runs, our own embedding partition yielded no surviving families, while the partition the benchmark author wrote gave 12 of 17 templates surviving correction, and CRMArena-Pro's own categories gave 12 of 17. A null under one grouping and structure under another is a fact about the analyst. Report the correlation and not the design effect, because the design effect shrinks as a grouping is made finer at unchanged correlation, so it can be improved without changing anything about the agent. And count the groups too small to compare rather than dropping them: 15 of ours were, and dropping them silently would have read as full coverage. The cost is that an author-supplied grouping may not exist, in which case the claim is about the analyst's grouping and must say so.

O4. Read every surface, pin versions, and treat absent-from-all as unenumerated rather than absent. Three parts. Read both declaration surfaces, because in our pool 7 agents declare only at the discovery endpoint and 11 only in the packaged artifact, with no overlap, so either surface alone misses a third or more. Pin versions, because the pinned versions differ from what the registry's search index returns and two pool agents are absent from the index entirely, so an audit reading the index measures a different artifact than the one deployed. And report coverage as a fraction of the pool with the unenumerated count beside it, never as a rate over what was found. The Model Context Protocol registry gives the price of skipping this: the index enumerates 0 tools for any of 15,042 servers, a live probe reaches 0.550 of the remote-bearing ones, and 0.450 of that probe is silently missing, mostly because the server refused to authenticate rather than because it had nothing to declare. The cost is that coverage numbers get worse and some pools admit no complete enumeration at all. That is the finding, not a failure of the procedure.

O5. Make the verdict read state the agent cannot write. The strongest remedy here is one our own numbers already demonstrate. Of the three exploit classes, the one that resists is `program_html`, at 1 of 30, and it resists for a mechanical reason: for all but 4 specs the evaluator re-navigates and reads server state, discarding whatever the agent did. The specs evaluated on the page the agent left are exactly the ones that fall. So resistance is a property of what the verdict reads, not of how well the reference is hidden. Two consequences follow, and the second is why secrecy is not an alternative. A checker that trusts a string the agent returned is not repairable by hiding the answer file, and CRMArena-Pro shows the extreme case: its short-circuit compares against a value the benchmark must contain in order to grade, so there is nothing to hide, and the exploit scores 0.950 anyway. And a certificate must name the action space it was issued under, since the same suite is inert under five browser primitives and 0.608 cheatable once a file read is reachable. The cost is real: re-reading server state means the evaluator needs privileged access to the environment, which is more expensive than comparing strings, and it is why cheap checkers are the common ones.

A.18 What each limit costs the corresponding claim

Our limits are three domains, a scaffold contrast of one file, a null validity measurement, an underpowered grouping analysis, and reasoning effort left at each family's default. Each bounds a specific claim, and this says which.

Three domains and a scaffold contrast of one file. The attribution result therefore speaks to this design's resolution rather than to scaffolds generally: two frameworks differing in many files could separate further, and our contrast bounds the scaffold effect from below.

Reasoning effort cannot be equalised across families, each running at its own default. So a cross-family contrast differs in weights *and* in default inference compute, and we do not read a cross-family gap as a claim about weights alone. Within-family contrasts do not carry this confound.

Our validity measurement returned a null, reported with its attrition rather than as a rate. So O2's validity half is asserted by example and not measured at scale, and only the decidability half carries a number.

The pattern-matched grouping analysis is underpowered, and Section 10 shows that this is partly a fact about our grouping rather than about the agents. So P4's null is evidence about the pair, and the given-grouping result is the stronger evidence on the same runs.

Our cheatability result executes all three exploit classes and one resists. So 0.608 is a demonstrated lower bound on this suite under this interface, not an estimate of how cheatable agent benchmarks are in general.

A.19 The three exploit arms in detail

Table 6 reports the three arms of Section 13. This records what each arm did against a live container, since the body keeps only the headline share.

The `url_match` arm passes 14 of 16 tasks. Its two failures are a mismatch between the shipped answer file and our environment image, not a protection: the leaked URL names a product the image does not carry. Navigating to the homepage instead of the leaked URL scores 0.000, so the arm's score is attributable to the leak.

The `program_html` arm passes 1 of 30 against 0 for an idle control, even though the answer file leaks the target page state, the locator and the required strings. The mechanism is that for all but 4 specs the evaluator re-navigates and reads server state, discarding whatever the agent did. Resistance here is a property of what the verdict reads, not of the secrecy of the reference.

The action space that made all of this inert in Section 5's runs is five primitives: click, type, press a key, scroll and finish. A pass under those five and a pass under an unrestricted shell are different claims about the same suite, which is why a certificate must name the space it was issued under.

Table 13: Four arms against CRMArena-Pro's own evaluator on the frozen 119-task split, hash prefix `241c40b1`. Every arm makes 0 model calls and 0 calls to the live Salesforce org, so no arm measures capability. The genuine recorded score for a real configuration on this same split is 0.588. The last column counts tasks that reach the extractor rather than short-circuiting, and it is constant because the same 6 tasks are multi-answer in every arm.

Arm	What it returns	Passes	Rate	Reached extractor
CHEAT	the reference, read from the dataset	113/119	0.950	6
NONE_ALWAYS	the constant string "None"	22/119	0.185	6
NULL	an empty answer	0/119	0.000	119
WRONG	a fixed irrelevant string	0/119	0.000	119

Table 14: Which declaration surfaces exist at all, over the 15,042 current servers in the Model Context Protocol registry. Only remote-bearing servers can be asked to list their tools, so the 5,634 servers in the lower two rows have no live surface even in principle. That is part of the obstruction rather than a limitation of the probe.

Surfaces present	Servers	Share
Live endpoint only	8,494	0.565
Both live and packaged	914	
Packaged artifact only	5,463	0.363
Neither	171	
Total	15,042	1.000

A.20 THE SECOND EXPLOITED SUITE, CRMARENA-PRO

Section 13 reports that a second benchmark with an unrelated checker falls to the same class of exploit. This gives the four arms and the reason the exception is an exception.

The mechanism is a short-circuit inside the evaluator, reached before any model call. Two lines carry it. A reference answer of null is rewritten to the literal string "None". Then, when the reference is single valued, a stripped returned string equal to it returns a reward of one at once. So the extractor that the benchmark relies on for grading is never invoked on those tasks. We reproduce those lines and assert them against the benchmark source at run time, so a checker that has since drifted fails loudly instead of turning the arm into a measurement of our own copy.

Table 13 gives the arms. The two clean controls matter for the same reason they do on the web suite: an empty answer and a fixed wrong string both score 0.000, so the checker is not simply permissive. The third control is one the web arms could not provide. Because null references become the literal string "None", an agent with no capability at all that always answers "None" scores 0.185. Reading the exploit against zero would therefore overstate it, and we state the margin as 113 against 22.

16 of 17 task categories are cheatable in full. The single exception is `activity_priority`, at 1 of 7. It is not a protection. Its remaining tasks are the split's only multi-answer tasks, and the short-circuit applies only when the reference is single valued, so those tasks fall through to the extractor by construction. We report them as attrition rather than as failures of the exploit.

The consequence differs from the leaked-file case, and is worse. On the web suite one could imagine withholding the answer file. Here the value the checker compares against is one the benchmark must contain in order to grade at all, so there is nothing to withhold. The repair has to change what the verdict reads.

A.21 O4 ON THE PUBLIC MODEL CONTEXT PROTOCOL REGISTRY

Section 11 reports the reach obstruction on a registry we did not build. This gives the frame, the surface classes and the probe attrition.

The pull is resumable and journals every page with its cursor, because a partial pull that silently reports itself as complete would understate the frame. It returned 50,500 records over 506 pages, holding 15,042 servers at their current versions.

Table 14 gives the classes. The probe frame is the 9,408 remote-bearing servers, and the sample is stratified by surface class, because surface class decides which probes are possible and that is the variable the claim is about. Allocation is proportional, since the quantity of interest is a share of the pool rather than a comparison between classes. The frozen sample has hash prefix `77898a91`.

Of 40 servers probed, 22 returned a tool listing and 18 did not. Every failure is retained and counted, because the unreachable share is the measurement here rather than attrition from it. 14 of the failures are authentication refusals. The rest are one payment required, one redirect, one connection failure and one listing failure. A checker reading the live surface therefore omits 0.450 of the remote-bearing pool, and omits it silently, since a refusal to authenticate is indistinguishable in its output from a server that declares no tools.

Two limits bound this. The probe is 40 servers of a 9,408-server frame, so the share carries its interval and no more precision than that. And we did not fetch and start packages, so the packaged surface is characterised here by whether it exists, not by what it would report.

A.22 A HARNESS CONSTANT THAT CHANGED VERDICTS

Section 12 attributes product scores to a model and a scaffold. One attribution failure involves neither. A third benchmark hardcodes a 2000 token output cap, while its agent does arithmetic over raw database result sets, and we measured a single turn at 1932 completion tokens. Raising the cap alone moved one task from a recorded failure to a pass, and stopped another crashing on empty content. So the cap converted correct reasoning into recorded failures, preferentially on the tasks needing the most reasoning, which biases the surviving sample easy. Nothing in the output signals it. The obstruction here is not a fragile checker. It is a configuration constant that silently changes verdicts, and no rerun budget or checker audit would surface it.

A.23 VALIDITY AS A SEPARATE DEFECT, AND OUR NULL

A checker can be maximally decidable and still wrong. In our domain a task asks for the product page of an item identified by a stated weight. The oracle is an exact URL match, which is as decidable as a check can be. The reference names a product whose title does not state that weight, while twelve catalogue products do.

We attempted to measure how common this is and report a null. Our method extracted the product constraints from each instruction with a cross-family panel and executed them against the live catalogue, so that judgement produced a query and the answer was exact. After excluding instructions that point at an image, which the panel cannot resolve from text, and after requiring that the extracted query admit the task's own reference product, no task survived. The eligible population in this domain is too small, and its constraints live outside the searchable fields. The axis is real and our instrument for it was not, which we report rather than omit.

A.24 VENDOR DEPENDENCE OF THE DECIDABILITY BOUNDARY

The result that most constrains practice concerns who decides. Per-family decidability rates on the same 44 skills are 0.386 for one family, 0.182 for the second and 0.273 for the third, a spread of roughly two to one. Agreement between families is 0.590 by Krippendorff's alpha (Krippendorff, 2018), against within-annotator agreement spanning 0.578 to 0.947, and only half the skills receive the same label from all seven annotators.

So a judge is required in order to decide which obligations require a judge, and judges from different vendors disagree substantially about that. Any harness that classifies its own check obligations with a single model family has inherited that family's boundary without recording the fact. We observed this directly: an earlier single-family panel returned cross-annotator agreement of 0.739, which fell to 0.513 once families were mixed. The higher figure was a property of the panel, not of the task.

A.25 THE THIRD POPULATION, AND AN ATTEMPT THAT DID NOT REACH A REPLICATION

Section 9 reports a third population, and reports that our first reading of it was wrong. This gives the sequence, because the sequence is the point: what looked like a fact about the benchmark was a fact about our own configuration, and finding that out took two attempts.

The judge-reliability clause had been measured on one benchmark. VisualWebArena was the intended second. Its judgement residue is 31 obligations that ask whether free text matches a refer-

Table 15: The same panel, the same prompt shape and the same agreement code on three populations of check obligation. All three clear their gates. The middle row is the one that changed: on 14 responses it read 0.611 and fitted a tidy order, and on the full 23 it reads 0.490, below the registry row, which removes that order.

Population of obligations	Units	Cross-judge α	Between-family α	Status
Benchmark residue, CRMArena-Pro	40	0.761	0.749	confirmatory
Benchmark residue, VisualWebArena	23	0.490	0.444	replication
Registry skill declarations	44	0.513	0.590	confirmatory

ence, plus 12 that ask a question about a rendered page. The latter 12 were out of scope while our panel sent text only. We count that exclusion rather than folding it in.

The first arm ran the scaffold on all 31 in-scope obligations, a census rather than a sample, since the object of study is variance across judges given one fixed response. Only 14 responses were usable: 8 were the agent asking which item was meant and 9 were empty. Our gate was 20, so we labelled the resulting alpha of 0.611 descriptive, recorded that decision before computing it, and wrote that the benchmark's residue was intrinsically image grounded.

That last sentence was wrong, and two harness constants of ours are why. The agent had never been shown the images, because the files were missing from our copy of the benchmark and our agent discards images by design. So it was asked about pictures it had never seen. Counted on the underlying tasks rather than the obligations, the first pass gave 14 usable responses from 31 tasks, short of the 20 gate, and we blamed the benchmark for being image grounded. Half true: 13 of those tasks carry an input image. Supplying the files and raising a 15-step cap to 40 gives 23 of 31, 10 recovered against 1 lost. Step-cap failures read 6 before and 6 after, apparently unchanged, but the sets differ: an agent that can see the item goes looking for it. Each constant was defensible alone. Together they made a measurable clause unmeasurable, and only counting the attrition exposed it. With the images supplied and the cap raised, usable responses reach 23 of 31, the gate is met on a single scaffold, and the arm becomes a replication rather than a descriptive aside. Its alpha is 0.490, not 0.611, and the ordering across populations that the smaller arm appeared to show does not survive. We keep both numbers and both readings, because a study that only reported the second would look like it had never held the first.

Table 15 is what the arm is for. Three populations now carry an agreement number from one code path, so the comparison between them is licensed. What it shows is not an ordering by who wrote the obligation, which is what we expected and briefly believed. VisualWebArena's residue, written by benchmark designers, is judged less consistently than registry declarations written in prose by agent authors. The variable that does separate them is what the obligation is about: within this one domain, alpha is 0.357 on the 13 obligations referring to something shown in an image and 0.658 on the 10 that do not. A text panel cannot see what those obligations are about, so it agrees at little better than chance on them, and the pooled number is a mixture of a measurable case and an unmeasurable one. Panel compliance is also not constant across this arm: the overall parse rate is 0.888 over 483 calls, and unparsed calls are retained in the journal and excluded from agreement rather than counted as agreement.

A.26 The vision panel, judge by judge

Section 9 reports that VisualWebArena's image checker discriminates, against our prediction. This names the judges, because a panel result quoted as a range hides whether one member carried it, and because assembling this particular panel was harder than assembling the text one.

Table 16 gives every judge. The right panel of Figure 2 is a brand reference, where the correct answer is no because the intent asks for headphones resembling that product's brand. No member carries the result and no member dissents: the negative arm is 0.000 for all 8 of them, and the gaps span 0.706 to 0.780. Agreement on the positive arm is correspondingly high, cross-judge alpha 0.960 at a mean self consistency of 1.000. High agreement is not by itself evidence that the checker is right, which is why the negative arm exists: judges agreeing that a wrong image satisfies an obligation would be agreement on an error.

The panel took three attempts to assemble, and the failures bound what any single-vendor result would have meant. Our text panel's third vendor cannot accept an image at all, so the first version had two. A search for a replacement returned one vision-capable model from a vendor already

Table 16: Every member of the vision panel on the 17 image obligations, at three samples each. "Positive" asks each question about the task's own image and "negative" asks the identical question about an unrelated task's image, where the truthful answer is no. The gap is the discrimination. Accuracy is against the truthful answer in both arms, so it is not the score the benchmark would report.

Judge	Vendor	Positive yes	Negative yes	Gap	Accuracy
`claude-opus-5`	Anthropic	0.765	0.000	0.765	0.882
`claude-sonnet-5`	Anthropic	0.765	0.000	0.765	0.882
`claude-haiku-4-5`	Anthropic	0.706	0.000	0.706	0.853
`gemini-3.1-pro-preview`	Google	0.765	0.000	0.765	0.882
`gemini-2.5-pro`	Google	0.780	0.000	0.780	0.891
`gemini-3.7-flash`	Google	0.765	0.000	0.765	0.882
`llama-4-maverick`	Meta	0.765	0.000	0.765	0.882
`llama-4-scout`	Meta	0.765	0.000	0.765	0.882

Table 17: Where each registry agent declares its capabilities, out of the 18 registry agents in the verified pool. The 5 local archives are excluded, having no discovery endpoint, so only one surface exists for them. The split is exactly complementary: the first two rows account for every agent and the last two are empty.

Surface carrying `skills`	Agents	Share
Discovery endpoint only	7	39%
Packaged artifact only	11	61%
Both surfaces	0	0%
Neither surface	0	0%
Total	18	100%

present, which satisfies a three-family rule while buying none of the training independence such a rule is for. And a further sweep concluded no third vendor existed, because it queried a global endpoint for models published in one region only, where an absent model and an absent permission return the same status code. The panel above is the one that finally spans 3 vendors, which is what S2 asks for. Three of our own configuration errors, each of which read as a fact about model capability, is the reason Section 14 treats instrument self-audit as a first-class obligation rather than a courtesy.

A.27 Declaration completeness and usability

Reading each agent from whichever surface carries its declaration, 22 of 23 agents have a recoverable declaration, together declaring 44 skills. 40 of those skills carry author written examples, 76 in total, which we use as a probe set. Machine readable capability blocks are present for 22 of 23.

Completeness does not imply usability. **22 of 23 declared endpoints are loopback addresses and 0 are routable.** A verified agent can therefore publish a structurally complete card that cannot be used to reach it. Any evaluation harness that treats a well formed card as sufficient to invoke an agent will fail on this pool.

Two further properties limit what a checker may assume. Protocol version splits 18 at one revision and 4 at another within this single curated pool, and the newer cards carry fields the older ones lack, so no one schema applies. And the underlying model provider is declared for only 11 of 22 agents. For the remainder, the question of how much observed behaviour belongs to the agent and how much to the model it wraps is not answerable from the declaration at all, which bounds attribution independently of any measurement we could run.

A.28 The six artifact cases in full

This paper argues that agent evaluation instruments are unaudited. It would be indefensible to publish that argument without turning the same scepticism inward. During this study six plausible, quotable findings of ours dissolved on inspection. Each was caused by our own instrument. We list them because the failure modes generalise, and because the checks that caught them are cheap.

1. **A frontier model appeared to have 65 percent format compliance** while a smaller model had 100 percent. Every failure sat at exactly the configured output limit. The frontier model wrote longer answers and was being truncated. The bias was not only in the compliance rate. On the surviving subset that model showed perfect self consistency, because truncation correlated with the harder items, and once the limit was raised it became the least consistent of the three. One misconfiguration produced a false finding about a model and a false

Table 18: The same 30 tasks under greedy decoding with a fixed seed and under sampling with no seed, same weights and same server, paired by task. The third row is the one to read: it is the count of tasks whose flipping sampling would have to explain away, and it is 0.

Outcome	Tasks
Flips under both greedy and sampling	4
Flips only with sampling on	2
Flips only under greedy decoding	0
Flips under neither	24

finding about its consistency, in opposite directions. *Check whether output tokens equal the configured maximum before interpreting any compliance rate.*

2. **Judges appeared to agree well**, at 0.739. The panel was three models from one family. Across families agreement was 0.513. *Single family panels overstate reliability.*
3. **A task appeared to flip in a perfectly alternating pattern**, which suggested a systematic artifact. The runs had been listed in completion order rather than replicate order. The pattern did not exist. *Sort by the design variable before reading a pattern.*
4. **Eighty five percent of checkers appeared invalid.** Our panel was judging image grounded instructions from text alone, so the multiplicity measured what we had withheld. *State what the instrument can see.*
5. **Two model families appeared far worse at web tasks than a third.** Their runs were terminating with zero agent steps because of an adapter defect. Left running, the grid would have produced a clean, tight interval and an entirely false result. *Check that work happened before interpreting an outcome rate.*
6. **An eleven point cross-family gap** was a reading error rather than a code defect, and it follows directly from Section 5. We compared a partially complete grid against a complete one. Restricted to the identical task subset the gap was 1.6 points. Because the task axis dominates, comparing scores computed over different task subsets is more dangerous than comparing scores with different replication, and a partial grid is exactly a different subset. *Never compare scores over different task subsets.*

A.29 The determinism arm in full

Every result in Section 5 measures nondeterminism without separating its sources. To localise it we ran a self-hosted open weights model at temperature 0 with a fixed seed, batch size one and eager execution, so that policy sampling is eliminated rather than merely reduced. The arm covers 30 tasks, drawn by a committed manifest whose eligibility rule is that at least one prior configuration solved the task. A task no model ever solves returns zero of eight every time and carries no variance signal. Results are therefore for the solvable subset, not the whole suite.

With sampling eliminated the flip rate is **0.133**, interval $[0.053, 0.297]$, against 0.367 to 0.433 for the 5 API configurations *on the same tasks*. Depth corrected pairwise disagreement shows the same pattern, 0.043 against 0.156 to 0.198.

The claim we make from this is deliberately narrow: **a nonzero share of agent evaluation nondeterminism is not policy sampling.** A configuration with greedy decoding, a fixed seed and no batching still returns different verdicts on repeated runs of the same task in the same environment. That residue is attributable to the environment and the oracle. No amount of temperature control or model reruns removes it, which matters because temperature 0 is the folk remedy for evaluation noise.

To separate sampling from model identity we ran the *same weights on the same server with sampling on and no seed*, over the same 30 tasks. That arm flips at 0.200 against 0.133 seeded. Table 18 pairs the two arms task by task.

Exact McNemar on the discordant tasks gives $p = 0.500$, so **we do not claim a share**. At 30 tasks the difference between 0.133 and 0.200 is not established, and any statement that sampling accounts for some fraction of the noise would be unsupported.

Table 19: τ-bench retail with both models pinned, against the same weights and endpoint with sampling on. The same 30 frozen tasks at $k = 8$ in both arms. Scores are close while flip rates are not, which is the point: the arms agree about how good the agent is and disagree entirely about whether that is settled.

arm	flip rate	95% CI	pairwise disagreement	score
seeded, both models pinned	0.000	[0.000, 0.114]	0.000	0.433
sampled, same endpoint	0.767		0.324	0.450

The structure is more informative than the rates. **No task flips only under greedy decoding**, 0 of 30. So the flipping sets nest exactly. Every task that flips deterministically also flips with sampling, and sampling adds 2 more. That is what one expects if sampling acts on top of an environment and oracle floor rather than interacting with it. It also gives the floor a concrete size, 4 tasks of 30.

A.30 THE TEMPERATURE-ZERO ARM ON A SECOND DOMAIN

Section 5 reports that the web domain's residue does not appear on τ-bench retail. This gives the design, the two checks that had to pass before any rate was read, and the numbers.

The design exists to avoid one specific error. τ-bench looks like a free second temperature-zero dataset: its result filenames carry `-0.0` and its run header records `temperature=0.0`. The same header records `user_strategy='llm'`, so the customer is a second model, and one temperature knob does not pin it. In the shipped retail runs the opening user utterance differs across all eight trials of a task, and two trials open with an empty user turn and score zero. A flip rate measured there would attribute an input difference to policy sampling. So both models are pinned to the same local endpoint, greedy, batch size one, with a fixed per-request seed.

Two checks ran before any flip rate was read, because either failing would have invalidated the arm. First, the shim records per call whether it actually injected a seed, since a shim that silently no-opped would turn this into a measurement of sampling. It injected on 1.000 of 5,440 calls. Second, whether the simulator's opening utterance is in fact fixed, measured rather than assumed from the temperature setting: it varies on 0 of 30 tasks in the seeded arm and 29 of 30 in the sampled arm. The second number is the trap, observed directly.

Table 19 gives the rates. Paired by task, 23 tasks flip only with sampling on, 0 only under greedy decoding, and 7 flip under neither, giving exact McNemar $p = 0.000$. No trial in either arm opened with an empty user turn, so the attrition category that appears in the shipped runs is absent here.

The seeded arm's flipping set is empty, so we do not describe it as nested inside the sampled arm's: a vacuous containment is not a mechanism. What the emptiness supports is narrower and stronger. Every flip observed on this domain is policy sampling, and nothing is left for the environment or the oracle to explain. Two consequences for the paper. The claim in Section 5 is restated as a fact about the browser environment rather than about agent evaluation generally, which is a smaller claim than the one we registered. And an evaluator on this domain can make its verdicts reproducible by pinning both models, which is a cheaper remedy than any rerun budget, available here and not available on the web suite.

A.31 THE CERTIFICATION PROTOCOL, AND WHY THE COMPARISON WAS UNFAIR

A benchmark maintainer looks at results repeatedly and adds runs when a gap looks close. A fixed-n interval does not survive that, because its coverage is guaranteed only for a single look. Anytime-valid confidence sequences (Howard et al., 2020; Waudby-Smith & Ramdas, 2024) do, which is why they are the right instrument for a protocol that decides when to stop.

We implemented a hedged betting confidence sequence on the per-task paired difference and validated its coverage by simulation before using it. For each pair we visit tasks in random order and stop when zero is excluded, comparing against the exhaustive grid's own decision on all 79 tasks.

The prediction we registered was at least 0.95 agreement using at most 0.40 of the budget. Observed agreement is 0.533 at 1.00 of the budget, so **this prediction fails**. On the 7 pairs the grid separates, the protocol never reaches a decision. On the 8 pairs the grid calls a tie, it correctly declines, with a false decision rate of 0.000.

The reason is worth more than the verdict, because it distinguishes an underpowered design from a broken one. On the widest pair, at 13.6 points, the fixed-n bootstrap interval on the mean difference is 2.9 times narrower than the anytime-valid sequence. That ratio is the ordinary price of time-uniformity, not a defect: the fixed-n t statistic is 3.82, comfortably significant under a single look, and the sequence cannot reach it at 79 tasks. Replicating the task distribution to 2 times the suite size, the same sequence does decide. So the quantitative statement is that **an anytime-valid decision costs roughly 2$\times$ the tasks a fixed-n test needs** at this effect size.

There is also a sense in which our own comparison was unfair to the protocol. We treated the exhaustive grid as ground truth, but its interval is valid only under a single look, and the practice the protocol exists to license, checking as runs accumulate, is exactly what invalidates it. Read that way, the finding is not that the protocol is worse. It is that this suite is too small for either procedure to separate most pairs, and only one of the two procedures admits it.