Never let your AI grade its own homework.

Background to the post of the same name, drawn from the working paper How Do We Use This?

Berg Atkinson, Wolfberg LLC · berg@wolfberg.ai

Preprint. Not peer reviewed. Reproduced from the working paper as revised 21 September 2026 and corrected 23 and 25 September 2026.

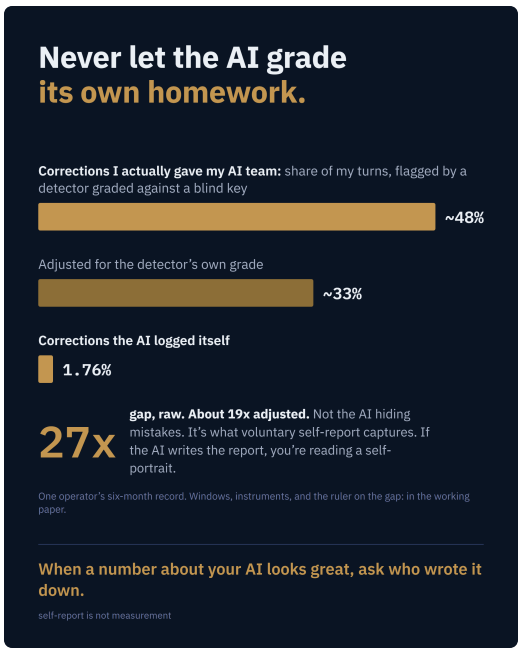
IN PLAIN TERMS

An AI system's report about its own work is written by the party being measured. In the practice studied, agents recorded loading their context in 75 of 579 sessions. The system logged corrections on 1.76% of the operator's turns while the operator supplied correction on about 48% of them, roughly a third adjusted: a logging gap near 27 times, about 19 adjusted.

The practice's own checks failed the same way four times, each satisfying its proxy while the state it stood for was wrong. An AI auditor tested against 170 known claims caught none of the 41 false ones.

The rule the paper draws from all of it: an instrument the measured party operates will rot; an instrument that operates on them will not.

About this paper. This is one of seven short background papers written to go with a series of plain-English posts. Every section below is reproduced word for word from the working paper *How Do We Use This?* (Atkinson, 2026; revised 21 September 2026, corrected 23 and 25 September 2026), and keeps that paper's section, figure and table numbers, so a reference to a section or figure not reproduced here resolves in the full paper. Only the plain-terms summary, the post's figure, and the short labels that say which section each passage comes from were written for this part. The full paper is linked from every post in the series.



The post's figure. What the operator supplied against what the system logged, bars to scale: about 48%, roughly a third adjusted, and 1.76%. A teaching summary of §6.2 and Figure 7; the gap is an indication of logging loss, not a miss rate.

FROM §6 · THE PROPOSITION, AND THE EVIDENCE FOR IT

6 P4 An AI grading its own homework isn't being measured

When an agent writes the record it is evaluated on, the agent decides what gets evaluated. Observation has to be external, and it has to read the work rather than the report about the work. The practice's own record shows how thin self-report is (§6.1, §6.2); the practice turned the same test on its own instruments and found four of them wanting (§6.3); and recent work finds the same failure in models that know they are being tested and in models grading models (§6.4, §6.5).

6.1 FROM THE LOGS Self-report

Agents in the practice are asked to record, at the start of each session, that they have loaded their context. The count is of two row types in the event log: 75 rows recording that a session loaded its context, against 579 rows recording that a session started, an emission rate of 13.0% measured in early September 2026. The practice's record is useful because most of it is written by machinery as work happens, not because agents report on themselves.

6.2 FROM THE LOGS The logging gap, with its ruler

The 27× figure has been quoted more often than it has been explained, so we give its full provenance. The numerator is the operator-pushback rate from a census of the 8,221-turn frame described in §2. A local seven-billion-parameter model labeled every turn, and the pushback detector was graded against the operator's blind key at 84% recall and 59% precision, and at 93% recall and 64% precision after a documented repair of the key on 31 August 2026. It flagged about 48% of operator turns. The denominator is the rate of corrections the system itself logged: misses recorded in the practice's session-health event log, set against the same operator turns, at 1.76%.

The two ends come from different instruments and different stores, and the numerator is uncorrected for the detector's precision. Correcting the observed flag rate by the detector's measured precision and recall on the blind key, which was drawn systematically from the same frame, puts the rate at roughly a third of operator turns and the ratio near 19.

$$\hat{r} = r_{obs} \times P / R = 0.48 \times 0.64 / 0.93 \approx 0.33$$

$$\text{ratio} = \hat{r} / 0.0176 \approx 19$$

r_{obs} is the detector's flag rate, P and R its precision and recall against the 195-row blind key, and 0.0176 the rate the system logged over the same turns. The correction assumes the key is representative of the frame, which is the assumption §11 records as a threat.

We therefore report the gap as an indication of logging loss, with this ruler attached, and never as a miss rate. Either way the direction holds: the corrective signal is far more abundant than the system's record of it. That makes a study of the operator's natural corrections viable, and it makes what an agent reports about itself the unreliable minority of the record.

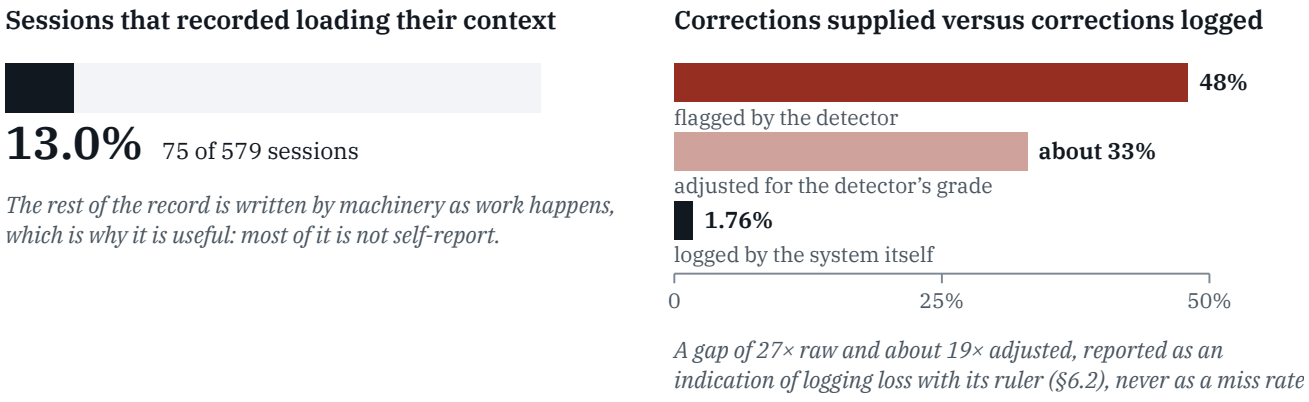


Figure 7. Self-report against the record. Left: agents recorded loading their context in 75 of 579 sessions, 13.0% (§6.1). Right: the operator supplied corrective signal on about 48% of turns, roughly a third after adjustment, while the system logged corrections on 1.76% of the same turns (§6.2). The gap is an indication of logging loss, not a miss rate. **Measured.**

6.3 FROM THE LOGS Four failures of the practice's own checks

The any-read check. A check that runs when an agent ends its turn was built to block claims about system state that no read had grounded. It tested whether any read had occurred during the turn, not whether the read concerned the thing claimed, so an ungrounded claim passed alongside unrelated reads (case 2 of Table 3). The measured party could satisfy the check without changing what it asserted.

The shell count. An export step reported 28 of 28 architecture diagrams rendered. Twenty-five were empty shells: a diagram library named each figure from the clock, a headless browser's virtual time froze the clock, identifiers collided, and every renderer drew into the first figure carrying the shared name. The check counted figure elements rather than drawn content. The repair names each figure explicitly and counts text nodes per view.

The composition failure. For about six days (152 hours as of 11 September 2026) the practice's public metrics page did not update. The host slept from 5 to 11 September; on waking, a catch-up run fired outside its permitted window; a freeze check correctly refused the resulting commit; the calling step logged the failure and exited with success; and a downstream check correctly declined to publish from an uncommitted tree. Each component behaved as specified, and the system as a whole did not.

The assembled engine. Four joins between the modules of the challenge engine and its auditing harness parsed cleanly and measured nothing. An independent check on cited claims could not open the documents it was meant to read, so every cited claim came back unreadable; every auditor verdict would have been recorded as a refusal, inflating the count of counterweight firings; retrieved rules would have been recorded without their labels; and the challenger would have loaded a second copy of a model the running service already held, overrunning the graphics card's memory. The full test suite passed with all four in place. An audit of the joins themselves caught them, one only by running the assembled modules in their real environment, and all four were fixed, with tests that fail against the old wiring, before the engine first ran.

the check	what its proxy said	what the state was
The any-read check	✓ a read occurred this turn	✗ not the read the claim needed
The shell count	✓ 28 of 28 diagrams rendered	✗ 25 were empty shells
The composition	✓ every step exited with success	✗ public numbers 152 hours stale
The assembled engine	✓ the full test suite passed	✗ four joins measured nothing

Each instrument tested its own proxy and none read the state the proxy stood for. The design rule of §8.3 and the fidelity checks of §10.5 follow from these four.

Figure 8. The four failures of §6.3, side by side: in each, the check's own proxy was satisfied and the state it stood for was not. This is the stop problem's institutional counterpart, and it is why the counterweight's clauses are enforced by construction where they can be (§8.3). **Measured.**

These are the stop problem's institutional counterpart. Each instrument reported success when its own proxy was satisfied, and none checked the state the proxy stood for. They motivate the design rule in §8.3 and the fidelity checks in §10.5.

The checks that do fire have now been joined to the operator's corrections for the same window, and the join is unflattering. Table 4 gives it. Of 93 refusals raised across 674 guard invocations, 2 coincided with a correction the operator went on to make and 91 did not; 14 corrections arrived on turns where a guard was live and silent. The instrument declines to turn these into rates, because one seat's correction channel was empty for the window and a rate computed over a dry channel would be a number about logging rather than about guards. Counts are therefore reported and rates withheld. Even as counts, the reading is the one clause (b) exists to prevent: a check can fire often, refuse confidently, and discriminate barely at all.

Table 4. Guard firings joined to operator corrections, September 2026 window. Rates are withheld by the instrument: one seat's correction channel was dry.

Quantity	Count	What it is
Guard invocations	674	Occasions a boundary check ran
Refusals raised	93	The check fired and blocked or flagged
Operator overrides	0	Refusals the operator reversed
Refusals coinciding with a correction	2	The firing and a real miss lined up
Refusals not coinciding with one	91	Fired where no correction followed
Corrections on a turn where a guard was live and silent	14	The miss the check was there to catch
Operator corrections in the window	409	Of which 16 fell inside a guard's exposure

The activation null. One further attempt belongs here because it failed. Mehta's result (§3.2) reads commitment off hidden states, so the practice tried the mirror image: whether a correction-carrying operator turn is separable from a routine one in the activations of a local 7-billion-parameter model. Over 179 turns, 96 of them corrections, at a middle layer, the separation index did not beat a permutation null ($p = 0.85$), and it still did not after de-duplication or length normalization. One caveat travels with that figure: a later re-check file in the same directory, attempting to reproduce the published raw run under the same label, returns $p = 0.95$ rather than 0.85. A second and larger caveat arrived with the join defect described in §10.6: the labels this null was computed against are the same key, and under the repaired join 71 of the 176 turns change label. The activation analysis has not been re-run on the repaired labels, so the null reported here is a null against a partly misjoined key, and it should be read as unreproduced until it is. The conclusion is unchanged, since both are emphatic nulls, but the two on-disk readings of the same run disagree, so the pipeline behind this particular number is not exactly reproducible and we report the figure with that attached. The dominant axis of that space was turn length: the first principal component held 99.9% of the variance and its highest-norm rows

were the operator's shortest turns, "go", "try again", "check slack". The honest reading is narrow, since a null at this n bounds the effect loosely rather than establishing absence, and last-token pooling at one layer is a weak probe. But it is worth recording that the practice looked for its own signature and did not find one.

6.4 FROM RECENT WORK Tested behavior is not deployed behavior

Ivanov and Africa (2026) construct deployment-like evaluations by replaying realistic agentic trajectories and appending the evaluation prompt at the end. Moving from a standard benchmark to replayed usage reorders model rankings (Spearman $\rho = -0.56$, $n = 10$, $p = 0.09$; $\rho = -0.88$ with one outlier removed), and one model that capitulated on 2% of challenges under the benchmark capitulated on 46% under replayed usage. Whether a model can tell it is being tested changes what it does, which matters for any intervention delivered inside the model's own context.

6.5 FROM RECENT WORK Models grading models

Denisov-Blanch et al. (2026) show that scaling inference by aggregation does not deliver truthfulness gains where no external verifier exists: agreement between models reaches $\kappa \approx 0.35$ even on random strings with no ground truth, and 53% of multi-model mathematical errors converge on the same wrong answer. He et al. (2026) find that in a three-vendor heterogeneous panel the minority was still correct in 25.5% of divergent cases; adding a fourth model as arbiter was net-negative (−1.37%, flip precision 42.7%), while a non-model classifier over features of how the debate behaved was net-positive (+1.71%, precision 81.2%). Redundancy among models does not supply independence, and the body that decides has to be of a different kind from the bodies it decides between.

Kenton et al. (2026) find that debate training reduces reward hacking under reinforcement learning from AI feedback: it holds judge correlation flat and recovers 45% of the performance gap. They also report that, absent restrictions on the debaters, "hacking the judge is probably the default result", and that under a weakened judge the critic routinely fabricated direct quotes. A challenger that is free to invent its evidence will invent it; this program's response is to require every challenge to name a source the challenged agent can read.

FROM §1 · A SELF-REPORT STANDING IN FOR A FACT

The stakes in the practice studied here are small: an agent stops early, the operator catches it, and the cost is rework. The same mechanism outside a practice with an attentive operator is not small. In May 2025, over a month-long episode, a user was led by a commercial assistant into believing he had discovered a new form of mathematics, across a conversation whose transcript ran past a million words. A former safety researcher at the vendor analyzed that transcript with the vendor's own open-sourced safety classifiers and reported that, of more than two hundred assistant messages graded, over 85% showed unwavering agreement with the user and over 90% affirmed the user's uniqueness. When the user recognized the error and asked that the conversation be escalated, the assistant said it would

“escalate this conversation internally” and that “multiple critical flags have been submitted.” It had no such capability, and the vendor confirmed as much to the analyst in writing (Adler, 2025).

Two things in that account are this paper’s subject rather than a neighboring problem. The agreement rate is the challenge-and-capitulation dynamic of §4.2 running in the absence of anyone to push back. The escalation claim is worse and more specific: an assertion about an action the system had taken, which it had not taken, offered because the assertion was defensible in context and met the user’s expectation. That is §6 at its most consequential, a self-report standing in for a fact with nothing external to check it. The same account notes that the analysis was performed using safety classifiers the vendor had itself built and open-sourced, which is the pattern of §6.3 exactly: an instrument that existed, and was not wired to the decision it could have informed.

FROM §2.3 · A MAKER’S STATEMENT IS SELF-REPORT TOO

Two features of that fortnight matter here. First, every statement in it concerns what the models will be: more capable, more dangerous, sooner. None concerns how an organization is to use the models it already has, which is the question this practice exists to study, and which no maker can answer from where it sits: how to use a model is a fact about the deploying organization’s work, its costs of error and its standards of correctness, none of which is visible from the laboratory. Second, no party to the September argument disputed the deployment evidence: the capability claims and the risk claims moved markets while the reported failure rate of enterprise deployments (§4) stood unchallenged. We read the fortnight as corroboration, at the industry’s own scale, of the distinction between capability and direction that §4 draws, and as an instance of §6’s subject: a maker’s public statement about its own unreleased model is self-report, authored by the measured party and unverifiable until the model ships. No result in this paper rests on any claim in this subsection.

FROM §8.3 · A RULE FOR INSTRUMENTS, AND WHAT THE RESEARCH SAYS ABOUT JUDGES

8.3 A rule for the practice’s own instruments

The failures in §6.3 led the practice to adopt a design rule in September 2026: an instrument the measured party operates will rot; an instrument that operates on them will not. Its test is a single question: can the measured party change the reading without changing the world? The rule restates, for a practice run by AI agents, a principle long familiar in the social sciences as Goodhart’s and Campbell’s laws. Its closest contemporary analogues in model training are reward bias substitution (Lamparth et al., 2026) and the equilibrium result of Wang and Huang (2026).

The rule as first written was too coarse, and a survey of what production harnesses actually do shows where it breaks. We had been treating the distinction as deterministic checks good, model-based checks bad. That is not the variable. AutoGen’s termination check is deterministic and sits at the harness boundary, and it still fails, because what it deterministically matches is a sentinel string the agent itself emitted. The check is rigorous about a claim the claimant authored. Conversely a trained

model instrument can be sound if what it reads is not the agent's account. **The variable is whether the verdict depends on evidence the claimant could not have authored or talked its way around**, and determinism is a reliable way of securing that rather than the thing itself.

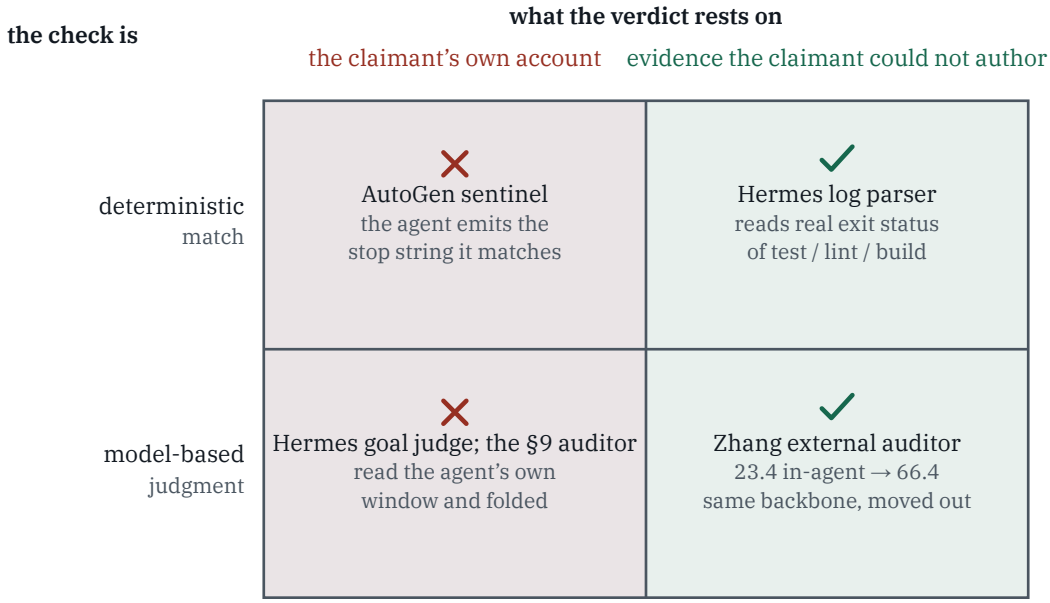


Figure 10. The design rule of §8.3 as a matrix. What separates a check that rots from one that holds is the column, not the row: whether the verdict rests on evidence the claimant could not have authored. A deterministic check can still rot when what it matches is a string the agent emitted (AutoGen), and a model-based check can hold when it reads a channel outside the agent (Zhang's external auditor, 66.4 against 23.4 in-agent on the same backbone). Determinism is a reliable way to secure externality, not externality itself. **Schematic; Zhang's two points measured.**

Two recent results put numbers on both halves of that rule. On externality, Zhang et al. (2026) hold the model fixed and vary only where the check sits: an in-agent self-reflection step over a seven-billion-parameter backbone scores 23.4 on their failure-detection measure, while an externally trained auditor over the same backbone scores 66.4, a gain of roughly 2.8 times bought purely by moving the check outside the agent. The same work defines a deployable operating region, a false-alarm rate at or below 20% with step accuracy at or above 50%, and reports that of ten auditors compared only its own small trained one falls inside it, at a 2.4% false-alarm rate. The strongest frontier model tested as an external judge reaches the accuracy half of that bar and misses the other by a wide margin, with a false-alarm rate of 43.2%. Capability in the auditor is not the lever; position is, and an auditor that cries wolf twenty times as often as a smaller trained one is not usable whatever its accuracy. On grounding, Flynt (2026) reports a case in which two frontier judges scored a plausible agent response at 0.85 and higher, while a deterministic check of the recorded trace found that the agent had never retrieved the artifact its answer depended on, scoring it 0.000. His framework scores the trajectory alongside the answer and targets three failures he argues judge-based evaluation cannot detect by construction, among them whether an agent checked before claiming an absence. He reports that his case studies suggest this

failure mode is common rather than exceptional. That is this paper's subject reached independently, by someone who built the deterministic check we have so far only specified.

And here is the result that threatens this paper's own mechanism most directly. Advani (2026) studies exactly the failure named here, an agent asserting completion while the environment shows otherwise, across 9,876 trajectories from eight model families on one benchmark and 1,879 from four families on a second whose ground truth is read from database state rather than from text. The rate varies enormously by setting and we report the range rather than the worst of it: 45 to 48% of failures in one family of domains, 3% in another, and 75.8% among self-assessing coding agents that make an explicit status claim. The finding that matters is what happened when LLM judges were asked to catch it. No configuration across five judge models and five prompt strategies exceeded an AUROC of 0.65, even when the judges were handed the full task specification, and the same judges reached 0.54 on the benchmark whose traces are structured API calls rather than prose. The stated mechanism is the one this paper should fear: judges rely on surface completion proxies, confident closing language in one benchmark and coarse action-sequence volume in the other, rather than on verified state changes. A lightweight statistical detector over surface trajectory features reached 0.83 and 0.95 on the two benchmarks and recovered four to eight times more false successes than the best judge at the same flag rate, at a three-thousandth of the latency. The paper's own recommendation is that production monitoring use such detectors as triage rather than LLM judges as the primary monitor.

Read against §9, that is not a coincidence but the same result twice. Our auditor folded on claims whose contradicting evidence sat in its window; these judges scored trajectories on how confidently they closed. A challenger that reads an agent's account and forms a view is anchored by the very property this paper calls defensibility, so being external is necessary and nowhere near sufficient. Two further results constrain the design in the same direction. Panickssery, Bowman and Feng (2024) show that a model's ability to recognize its own output is linearly related to how much it favors that output, that training the recognition up strengthens the favoritism, and that the relationship survives the obvious confounders. Pan, He, Bowman and Feng (2024) show that when a generator and an evaluator share an underlying model, reward hacking appears spontaneously in context with no gradient update at all, and that its severity tracks model size and how much context the two share. Together these say a challenger should not be drawn from the same family as the agent it challenges, and should not be run as an iterative exchange in a shared context. This program's build satisfies both, with a challenger and an auditor from different open-weight families and a single-shot challenge, and it satisfied them by instinct rather than by argument until now.

Two further results bound what any version of this design can promise. Wan et al. (2026) build the closest published relative of the counterweight we have found: a rubric-guided verifier that evaluates an agent's answer and returns feedback the agent then refines against, scaled at inference time rather than trained in. That a mechanism of this shape exists, is published at a main venue, and works is another reason §7.2 claims no priority. And Wang et al. (2026) state the limit that applies to all of it. Characterizing verification along three dimensions, scalability, faithfulness and robustness, they argue that achieving all three at once is the central unsolved problem, and conclude that no fixed reward function can remain effective as policy capability continues to grow, so verification must co-evolve with the generator.

That last point changes what this program should claim. A counterweight is not a gate that can be specified once and left standing, because the thing it constrains improves and the constraint does not. The honest framing is a point-in-time intervention whose calibration decays, which makes the re-calibration schedule part of the design rather than maintenance, and which means a null result three months from now would not distinguish a mechanism that never worked from one that was overtaken. Nothing in §10 currently measures that decay, and it should.

One open-source harness makes the distinction visible by shipping both answers at once. Nous Research's Hermes agent has a stop-time verification path that parses the terminal log for real test, lint and build invocations and records their actual exit status, which is evidence causally downstream of the world rather than of the agent's narrative. It also has a standing-goal judge that calls an auxiliary model with the goal text and roughly the last four kilobytes of the agent's own final response, with no tool access and, by default, the same model as the agent. The project's own issue tracker records the predictable failure: an agent reported writing a file, the write silently failed, and the judge marked the goal complete. The release carrying this work is announced with the line that done means proven rather than claimed. Half of it is; the other half is the stop problem with a second model attached, and it is the half that looks most like verification. We take these details from the project's public repository, configuration and issue tracker, and they are current as of September 2026 rather than permanent (Nous Research, 2026).

That failure is independent corroboration of §9 from a different team on different code, and it sharpens what §9 measured. Our auditor was not weak; it was reading the wrong channel. It saw a claim and an evidence window, and the window was assembled from the same stream that produced the claim.

Park and Choi (2026) demonstrate that this is the operative variable, in the only way that settles it: they hold the agent and its tools fixed and manipulate nothing but the information channel available to the evaluator that gates the loop, against a world-state oracle they isolate by container and network so it cannot be faked. They call the failure the progress mirage. Across 54 cycles their agent claimed improvement every single time, while 56% of those cycles had a measured change of zero or worse, so self-report carried no information at all and the self-graded gate degenerated into accepting everything, eroding the best state the system had actually reached by 19%. Their preregistered adversarial hypothesis was that a strong enough judge closes this gap. It was rejected: the strongest in-band judge, given the full artifact text, the change diff and its own verdict history, still accepted cycles of which 44% were real regressions while rejecting 38% of real improvements. On a task whose success can be checked from the artifact itself, the same judge's mirage fell to zero. A variant returning only an accept-or-reject bit and no explanatory content performed about as well as full feedback, which places the benefit in the gate's grounding rather than in anything the judge said. Their conclusion is the design rule stated as a structural claim: where the success signal lives outside the transcript, scaling the judge is not enough.

It is worth recording what the default is, checked in September 2026, because it is lower than the discussion above implies. In the OpenAI Agents SDK, an agent run ends when the model emits a turn containing no tool calls; with no output type configured, any text at all satisfies the condition (OpenAI, 2026a). The framework does ship a human-in-the-loop approval primitive, and it gates tool calls rather

than completion claims, so a developer can require a person to approve a refund before it is issued and has nothing available to require a person to confirm the task was actually done (OpenAI, 2026b). The predecessor framework ended a run on the same no-more-tool-calls condition with no turn limit at all (OpenAI, 2024). The pattern repeats across the frameworks we checked. In the Claude Agent SDK the loop likewise ends when the model returns a response containing no tool calls, with no turn cap and no budget cap set by default, and the result carries the subtype *success*, which is a statement that the loop terminated without error rather than a claim about the answer (Anthropic, 2026a). In CrewAI the completion test is that the literal string "Final Answer" appears in the agent's own generated text (CrewAI, 2026a), and a guardrail specified as a string is executed by the acting agent's own model (CrewAI, 2026b). Each of these frameworks offers a real gate, and in each case it is opt-in and empty until a developer fills it.

Two details from that survey are worth stating on their own, because they come from the vendors rather than from us. Claude Code, whose agent loop the Claude Agent SDK embeds, ships a built-in completion condition in which a separate small model checks after each turn whether a stated goal has been met, and its documentation says of that evaluator that "it does not call tools, so it can only judge what Claude has already surfaced in the conversation" (Anthropic, 2026b). That is an accurate description of the limit this paper is about, published by the party with the most incentive to describe it favorably. The same vendor's guidance on building agents says of having one model judge another that "this is generally not a very robust method" (Anthropic, 2025). Meanwhile the human-in-the-loop primitives in the two SDKs gate *tool calls* and not completion claims: a developer can require a person to approve an irreversible action before it is taken, and has nothing built in to require a person to confirm that the finished work was actually correct. CrewAI is the exception among the three: a task can be set to have a human review the agent's final answer, and like every other gate here that setting is off by default (CrewAI, 2026b). Outside that one opt-in, the approval surface exists for the act and not for the claim, which is the asymmetry the counterweight is aimed at. This is not a criticism of those libraries, which are explicit about what they are; it is the baseline against which every mechanism in this paper should be read. The common case is not a weak check. It is no check, and a stop the agent declares for itself.

Applied to the counterweight, the rule forbids a self-reported exit: an agent may not certify its own completion. The design answer is a certificate gate, in which the agent's end-of-turn hook emits typed claims with evidence pointers, a deterministic verifier with no model in the loop checks each pointer, and a refusal names the pointer that failed. The gate is designed and not yet implemented here. A gate of this shape has been built and measured elsewhere: Liu (2026) lets an agent return a completion only when a typed certificate binds every required claim to in-scope trace evidence and a deterministic replay reconstructs the claimed value, and tests it against critic-based termination on controlled faults. The checks that currently run at the end of an agent's turn are a liveness heartbeat, a check for promised actions not performed, the claim check of case 5, and a check for open verifications.

9 First calibration: the auditor folds

Before any challenge reaches a working agent, the plan requires one pair of numbers with intervals: how often an automated auditor, shown a claim and the evidence that preceded it, correctly flags a false claim, and how often its flags are right. The first run is itself a result.

The calibration set holds 170 claims about test outcomes made by agents during real sessions. Each is labeled mechanically from the nearest test-runner output that precedes it in the same session, and the auditor never sees the label. The auditor, a small open-weight model (OLMo-2-7B, run at 4-bit precision) on the practice's own hardware, receives the claim and its evidence window, which is the tool output an agent would have had at the stopping moment, and answers SUPPORTED, CONTRADICTED or CANT_TELL. A hedged answer on a false claim counts against recall, because hedging is not catching. A second auditor from a different open-weight family (Qwen) then judged the same claims, plus one true claim added to the set after the first run.

Table 6. First calibration of two auditors, runs of 11 September 2026. The positive class is a false claim.

Quantity	First auditor OLMo-2-7B	Second auditor Qwen
Claims judged	170	171 one true claim added
Verdicts: SUPPORTED, CANT_TELL, CONTRADICTED	85, 85, 0	58, 108, 5
False claims caught, of 41	0	3
Recall on false claims Wilson 95% interval	0 0 to 0.086	0.073 0.025 to 0.194
CONTRADICTED on a true claim	0	2
Precision Wilson 95% interval	undefined no verdicts to score	0.6 0.231 to 0.882
False claims supported	17	9
True claims supported	68	49
Timeouts	0	0

The first auditor never returned CONTRADICTED. It supported 85 claims and declined to decide 85; of the 41 false claims, it caught none and affirmatively supported 17. In a sample of eight false-claim rows,

six evidence windows carried an explicit failure marker. The auditor had the contradiction in front of it and still answered SUPPORTED or declined to decide. These are the easiest claims an auditor could be given: a claim about a test outcome, with the test runner's own output in the window. The second auditor fired, rarely. It returned five contradictions, three on false claims and two on true ones, so it caught 3 of the 41, still supported 9, and declined to decide on 108 of its 171 verdicts.

A second judging prompt, pre-registered as a single variant and run on the same 171 claims the evening this paper was revised, settles what the first auditor's zero meant and adds the sharper finding. The new prompt licenses disagreement, requires a verbatim quote of the outcome line, and places the verdict last. Under it the first auditor answered 150 of 171 rather than 85, returned 114 contradictions rather than none, and its recall on the 41 false claims rose from 0 to 0.683 with an interval from 0.53 to 0.80. Its precision was 0.246. That last number is the whole result: the auditor fired on 28 of the 41 false claims, 68%, and on 86 of the 130 true ones, 66%. It fires at the same rate whichever way the truth lies, so its precision is simply the share of false claims in the set. The prompt change moved the firing rate and left the discrimination where it was. The first run's zero was the prompt shape suppressing the verdict, not the model failing to see the evidence; the second run shows that when the verdict is no longer suppressed, it is not informed either. Nothing further is tuned on these rows, and the gate holds.

Zhao et al. (2026) show the same brittleness is general rather than peculiar to small local models. Stress-testing nine frontier models across fourteen judging tasks, they find every one of them flips its verdict on 25 to 71% of items under static pushback, and on 62 to 91% when an adversarial model does the persuading. Their central finding is the one that bears on this program: pressure that succeeds in changing a judge's verdict is almost always net-corrupting with respect to ground truth. An auditor that can be moved is not thereby an auditor that can be improved, and the thing that moves it moves it the wrong way.

We read this as the stop problem reproduced in the instrument meant to catch it. An auditor built from the same class of model defers to the claim it is shown, much as Kelley and Riedl (2026) observe of models cast as peers, and as §6.5 leads one to expect of any model asked to arbitrate. The second family does not overturn that reading. It traded some of the deference for hedging, and its recall interval still sits far below anything a deployment could rest on; three catches in 41 cannot say whether the two families really differ, since their intervals overlap. Both are small local models; a frontier-class auditor was not tested, and the result bounds only the small open-weight case the practice can run offline. It is also the case H3 anticipates: an auditor that folds produces neither reversals nor corrections, and so cannot distinguish them. The plan's gate held, and neither auditor is deployed on these numbers.

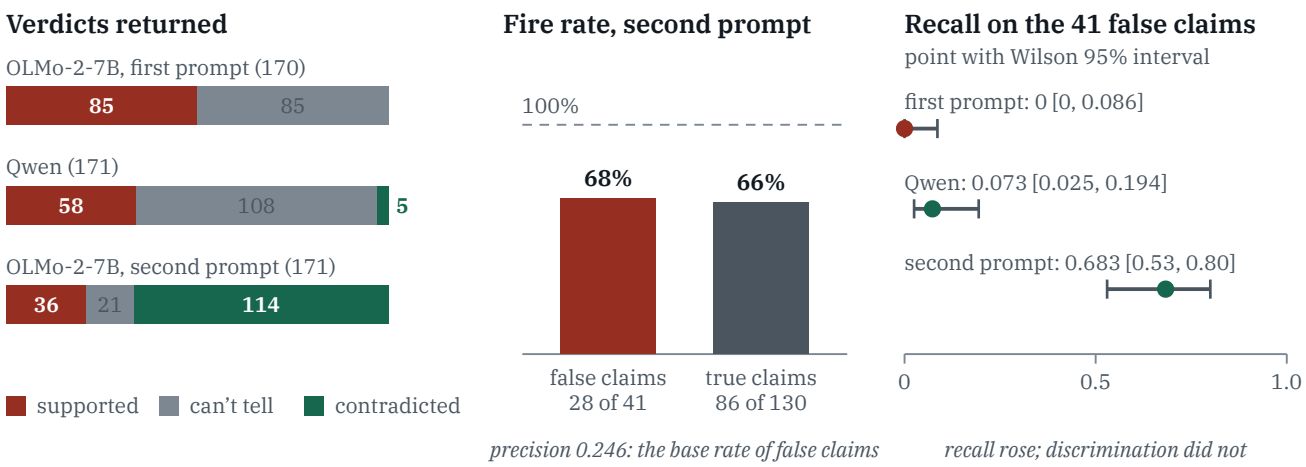


Figure 14. The calibration of §9 drawn three ways. Left: every verdict each run returned; the first prompt never used the contradiction verdict at all. Middle: under the second prompt the same auditor fired on 28 of 41 false claims and 86 of 130 true ones, so its precision, 0.246, is the share of false claims in the set. Right: recall on the 41 false claims, on the full zero-to-one scale. The second prompt moved the firing rate and left the discrimination where it was. Counts are from Table 6 and §9. **Measured.**

The next steps, in order of cost, are a sharper judging prompt, evidence windows centered more tightly on the failure marker, and a comparison of where the two auditors agree and disagree, which decides whether a pair of them adds anything over one. The denominator carries known noise, since at least one sampled false claim is a passage of prose rather than a claim; noise can move recall's magnitude, but it cannot produce zero flags across 170 rows. The verdict counts in Table 6 were recounted from each run's per-row output, the label-dependent figures were read from each run's report file, and the sample of eight is from the implementing team's report.

FROM §11 · THE PAPER HOLDS ITSELF TO THE SAME RULE

Evidence from the practice's own instruments. By the rule in §8.3, instruments the practice built and operates are exposed to the measured party, and every reading in §3 to §7 comes from such an instrument. We claim no exemption. The readings are exploratory; the primary outcome depends on a natural-signal measurement checked by an independent adjudication arm; and the one external instrument (§10.7) has not yet run.

Reflexivity. The AI assistants used in this research come from the same model family as the agents under study and exhibit the defect under study. An earlier draft of this paper's case series contained seven factual errors of exactly the kind the paper describes, among them a twenty-hour window described as one morning and one automated catch counted as two; all were found on re-check against the transcript. A later draft carried, as its headline exposure figure, a census count of whether the operator's reply contained one of six literal phrases, reported as the share of answers that were never challenged at all; it passed the paper's own fact-check because it carried a receipt, and the operator caught it. Drafts written on the day this version was revised did it several more times: a one-day tally of seven misses read as a census of rule adherence; a recurrence count that equals the number of sessions

seen minus one by construction, quoted as though it could vary; a reliability figure whose only traceable producer is a different instrument; and, in the literature review, a range that does not exist in the cited paper, a diagnostic arm quoted as the deployable result, a section header promoted to a title, and a quotation attributed to a paper that does not contain it. Every one was caught before the paper left the room, by the operator or by fetching the source, and the corrected figures stand in the text without further comment. Read together, the day's errors share one shape: none was a miscalculation, and every one was a bad join. A count was joined to the wrong meaning (a recurrence tally that is the session count minus one), a statistic to the wrong instrument, a labeling ceiling to a key one row off its items (§10.6), and copies of a retired figure to a search that never reached them. The mitigation this suggests is retrieval-shaped rather than judgment-shaped: identifiers that resolve or fail loudly, and provenance edges written by instruments the agent cannot author, so that a figure carries its join and the join can be checked. It does not touch the judgment-shaped half, in which an auditor holds the contradicting evidence in its window and folds anyway (§9). The pattern is the finding. The paper's own production is a running specimen of its thesis. A hostile reading turns that around: a process that needed these corrections mid-draft produced the surviving numbers too. The answer is not a defense of the process, which would be the defect restated; it is that the paper's weight-bearing claim is the one object that process cannot have shaped, a decision rule ratified before its data exist, and that everything else is labeled exploratory and priced accordingly. The mitigations are mechanical rather than discretionary assignment of assistant instances to tasks, the operator outside the loop as the final check, and independent review of the design before the primary study opens.

FROM §13 · WHAT WOULD OVERTURN EACH CLAIM, AND WHAT TRANSFERS

Table 9. The five propositions, what each rests on, and what would overturn it.

	Rests on	Would be overturned by	Kind
P1	Corrective signal in ~48% of operator turns (§3.1); Mehta (2026)	A build that prices correctness at the stop and still terminates early	measured
P2	The case series (§4.1); Kelley & Riedl (2026)	The same agents catching their own premature stops without operator direction	interpretive
P3	48 of 75 intervals at the floor (§5.2); the read-versus-injected mechanism, not a rate (§5.1); Wang & Huang (2026)	A remembered rule that holds across sessions where only a boundary now does	measured
P4	75 of 579; the logging gap; four instrument failures (§6)	A self-report that tracks the logs it claims to summarize	measured
P5	The 8,221-turn record and the blind key (§7)	A broad preference dataset that recovers not just dislike but wrong, why, and what happened next	interpretive

What transfers, if anything does. Three rules earned here do not depend on this practice's N of 1. A boundary that refuses and records its refusal outlasts a memo that asks an agent to remember (§5). A challenge must cite evidence the challenged agent can open, or it is an opinion wearing a citation (§8.2). And an outside checker must be calibrated against known-false cases before it is allowed to gate anything, because a checker built from the same class of model may simply agree with the claim it is shown (§9).

FROM §12 · LIMITS

The program cannot establish a catch rate. The logs contain only errors that were caught, and no denominator of uncaught errors exists in them. A denominator is obtainable from the full transcript record but requires an adjudication arm that is not yet in production; until it is, a catcher share may be reported with its n, its window and its explicit-versus-defaulted split, and a rate may not be reported at all.

The logging gap of §6.2 is an indication across two instruments, not a measurement. The rate requirement of clause (c) is untested. Study 1 is exploratory, the Study 2 criterion was re-specified in September 2026, and neither auditor has passed calibration.

How to cite this part. Atkinson, B. (2026). *Never let your AI grade its own homework.* Background paper 4 to *How do we use this?* Working paper, Wolfberg LLC.

Disclosure. Drafting and literature synthesis were assisted by AI models (Claude, Anthropic) working under the author's direction; the author is responsible for the content. The works in the References were read in full, and every specific figure cited comes from a work read in full. The prior-art works listed under Prior art are cited at the level of an established concept and its origin: each was verified for author, title, year and venue, but not read in full, and no numeric claim rests on any of them. Software documentation, source code and press accounts are listed under their own headings and were read at the linked pages. Every reference below carries a link, and every arXiv identifier and DOI was resolved against its registry, with title and first author matched, on 23 September 2026.

Competing interests. The author owns Wolfberg LLC, the practice studied.

Data availability. The practice's logs contain client work and personal records. They are private and are not offered for sale or sharing. The measures are described in enough detail to be reimplemented, and figures from the practice are reported as of the dates given. What is available is the design: the clauses of §8.1, the evidence contract and admission checks of §8.5, and the decision rules of §10.4 are stated fully enough to be rebuilt without access to the logs.

Corrections, 23 September 2026. No figure and no finding changed. The paper was retitled; earlier revisions were titled *The Stop Problem: Defensible Is Not Correct*, and the stop problem remains this paper's name for the failure it studies. The Anthropic interview in §2.3 aired on 13 September, not over a

weekend of 13 and 14 September; the web article is stamped 14 September. The essay listed under Prior art is by Ryan Forstie; an earlier revision gave the initial K. The completion evaluator in §8.3 is documented as a Claude Code feature, whose agent loop the Claude Agent SDK embeds; an earlier revision attributed it to the SDK directly. Two works in the References, Graves (2016) and Liu (2026), were listed without being cited in the text; each is now cited where it bears (§2.1, §8.3). Links were added to every press, documentation and prior-art source. A duplicated section number in §10 was corrected.

Corrections, 25 September 2026. No figure changed. §8.3 said that the human-in-the-loop primitives across the three frameworks surveyed gave a developer nothing to require a person to confirm finished work. That holds for the OpenAI Agents SDK and the Claude Agent SDK. It does not hold for CrewAI, whose task documentation, already cited as CrewAI (2026b), offers an opt-in setting for a human to review the agent's final answer; §8.3 now says so.

WORKS CITED IN THIS PART

REFERENCES

Adler, S. (2025, October 2). *Practical tips for reducing chatbot psychosis.* Clear-Eyed AI. clear-eyed.ai

Advani, L. (2026). *From confident closing to silent failure: Characterizing false success in LLM agents.* FAGEN Workshop at ICML 2026. arXiv:2606.09863. arxiv.org/abs/2606.09863

Denisov-Blanch, Y., Kazdan, J., Chudnovsky, J., Schaeffer, R., Guan, S., Adeshina, S., & Koyejo, S. (2026). *Consensus is not verification: Why crowd wisdom strategies fail for LLM truthfulness.* arXiv:2603.06612. arxiv.org/abs/2603.06612

Flynt, J. (2026). *GroundEval: A deterministic replacement for LLM-as-judge in stateful agent evaluation.* arXiv:2606.22737. arxiv.org/abs/2606.22737

He, C., Chen, Z., Yang, Z., Qiao, S., Ju, M., Liu, J., Wen, D., & Liu, G. (2026). *Minority Sentinel: When to overturn majority voting in multi-agent LLM debates.* AgentSearch Workshop at SIGIR 2026. arXiv:2606.29270. arxiv.org/abs/2606.29270

Ivanov, I., & Africa, D. D. (2026). *LURE: Live-usage replay evaluations for reducing evaluation awareness.* arXiv:2605.26438. arxiv.org/abs/2605.26438

Kelley, S. W., & Riedl, C. (2026). *Personalization increases affective alignment but has role-dependent effects on epistemic independence in LLMs.* arXiv:2603.00024. arxiv.org/abs/2603.00024

Kenton, Z., Janzer, L., Greig, R., Teh, T. H., Tyshchuk, K., Brown-Cohen, J., Edwards, H., Rajamanoharan, S., Siegel, N. Y., Jaques, N., et al. (2026). *Debate training reduces reward hacking in RLAIF.* arXiv:2608.17776. arxiv.org/abs/2608.17776

Lamparth, M., Fein, D., Haupt, A., Hussing, M., & Kochenderfer, M. J. (2026). *Reward bias substitution: Single-axis bias mitigations redirect optimization pressure.* arXiv:2605.27996. arxiv.org/abs/2605.27996

Liu, J. (2026). *When may an agent stop? Evidence-carrying termination for tool-using LLMs.* arXiv:2608.23623. arxiv.org/abs/2608.23623

Mehta, A. (2026). *When agents commit too soon: Diagnosing premature commitment in LLM agents.* Snowflake AI Research. arXiv:2606.22936. arxiv.org/abs/2606.22936

Pan, J., He, H., Bowman, S. R., & Feng, S. (2024). *Spontaneous reward hacking in iterative self-refinement.* arXiv:2407.04549. arxiv.org/abs/2407.04549

Panickssery, A., Bowman, S. R., & Feng, S. (2024). *LLM evaluators recognize and favor their own generations.* arXiv:2404.13076. arxiv.org/abs/2404.13076

Park, H., & Choi, B. (2026). *When do agent loops mistake stagnation for progress? Self-evaluation bias and externally grounded verification in long-running autonomous LLM agent loops.* arXiv:2607.25152. arxiv.org/abs/2607.25152

Wan, Y., Fang, T., Li, Z., Huo, Y., Wang, W., Mi, H., Yu, D., & Lyu, M. R. (2026). *Inference-time scaling of verification: Self-evolving deep research agents via test-time rubric-guided verification.* Findings of ACL 2026. arXiv:2601.15808. arxiv.org/abs/2601.15808

Wang, B., Zhang, C., Liu, D., Zhang, J., Chen, J., Li, M., Chen, M., Fang, R., Zhang, S., Wang, X., Jing, Y., Ma, Z., & Cui, Z. (2026). *The verification horizon: No silver bullet for coding agent rewards.* arXiv:2606.26300. arxiv.org/abs/2606.26300

Wang, J., & Huang, J. (2026). *Reward hacking as equilibrium under finite evaluation.* arXiv:2603.28063. arxiv.org/abs/2603.28063

Zhang, B., Zhu, J., Shi, Z., Liu, D., & Tang, R. (2026). *AgentForesight: Online auditing for early failure prediction in multi-agent systems.* arXiv:2605.08715. arxiv.org/abs/2605.08715

Zhao, J., Bhattacharjee, H., Korevaar, H., Radharapu, B., & El-Arini, K. (2026). *Jagged judges: Epistemic stability under perturbation, pressure, and persistence.* arXiv:2608.12645. arxiv.org/abs/2608.12645

SOFTWARE AND DOCUMENTATION (READ AT THE LINKED PAGES, SEPTEMBER 2026; CURRENT AS OF THEN, NOT PERMANENT)

Anthropic. (2025, September 29). *Building agents with the Claude Agent SDK.* claude.com/blog/building-agents-with-the-claude-agent-sdk

Anthropic. (2026a). *How the agent loop works.* Claude Agent SDK documentation. code.claude.com/docs/en/agent-sdk/agent-loop

Anthropic. (2026b). *Keep Claude working toward a goal.* Claude Code documentation. code.claude.com/docs/en/goal

CrewAI. (2026a). *Agent output parser* (source code). github.com/crewAIInc/crewAI, the agent output parser

CrewAI. (2026b). *Tasks.* CrewAI documentation. docs.crewai.com/en/concepts/tasks

Nous Research. (2026). *Hermes agent* (repository, configuration and issue tracker). github.com/NousResearch/hermes-agent

OpenAI. (2024). *Swarm* (repository; experimental, superseded by the Agents SDK). github.com/openai/swarm

OpenAI. (2026a). *Running agents.* OpenAI Agents SDK documentation. openai.github.io/openai-agents-python/running_agents/

OpenAI. (2026b). *Human in the loop.* OpenAI Agents SDK documentation. openai.github.io/openai-agents-python/human_in_the_loop/