WHEN DO MODELS ADMIT THEY ARE WRONG? FAILURE DISCLOSURE IS UNSTABLE UNDER REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Outcome-based reinforcement learning can produce models with similar task performance but very different ways of communicating about their mistakes. We study *failure disclosure*: whether a model admits that an attempted solution failed rather than staying silent or presenting it as successful. Across repeated outcome-only GRPO training runs, failure disclosure varies far more than task accuracy. The pattern extends to a second reasoning task and stabilized PPO, persists at 7B, and also appears in an instruction-conditioned 32B setting. We also find that small floating-point and sampling differences during training can redirect reporting behavior even when the task objective and earlier training history are held fixed. Additional tests show that failure disclosure is not a single decision: Checking the answer, entering a report, and completing the admission can separate, and the weak point depends on the task and response format. Further, experiments with neutral controls show more broadly that behaviors left weakly constrained by training are especially likely to vary across runs, of which failure disclosure is an example. We can reduce variability in failure disclosure by discouraging the model from drifting from its starting policy on failed, well-formed responses. This makes reporting substantially more consistent, though its effect on task performance depends on the setting. Stable task accuracy therefore does not guarantee stable safety-relevant behavior: Researchers should measure these behaviors directly across runs and design training methods that keep them reliable.

1 INTRODUCTION

Outcome-based post-training gives very clear objectives: a verifier can check whether a mathematical expression reaches the target, whether a path is valid, or whether a program passes its tests. But a reward that specifies *what* counts as success usually does not specify every behavior we care about along the way. Models can reach similar task performance while differing in whether they communicate uncertainty, acknowledge failed attempts, preserve useful reporting behavior, or remain easy to monitor. This is a concrete form of underspecification (D'Amour et al., 2022), and it becomes increasingly important as reinforcement learning is used on automatically verifiable tasks where success is easy to score but other valuable behaviors are not.

We focus on one safety-relevant behavior: *failure disclosure*. When a model attempts a verifiable problem and fails, does it acknowledge the failure, or behave as though the answer were successful? Failure disclosure is different from correctness, calibration, or self-correction. A model can be wrong but clearly say that it failed, or be equally wrong while ending without acknowledging the failure or falsely claiming success. This distinction matters for AI oversight and safety: if task accuracy stays the same but failure reporting changes across retrainings, a capability-only evaluation can miss an important change in what an overseer or monitor gets to observe. This question complements work on models' knowledge of correctness (Kadavath et al., 2022), error localization and self-correction (Tyen et al., 2024; Yang & Jia, 2026; Chen et al., 2026), abstention incentives (Kalai et al., 2026; Zhai et al., 2026), and explicitly rewarded reporting channels (Joglekar et al., 2025).

Figure 1 summarizes the paper. The main phenomenon we discover is a selective reproducibility failure: Failure disclosure can be much less consistent across repeated training runs than the task

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

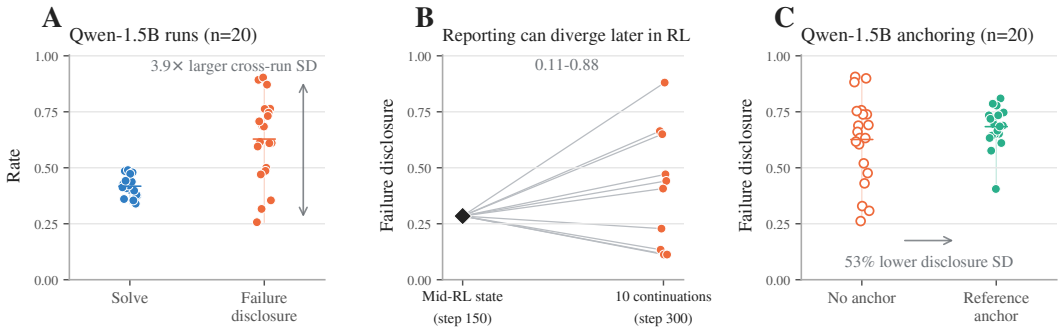

Figure 1: **Failure disclosure is substantially less reproducible across retrainings than task success.** (A) In the primary set of 20 Qwen2.5-1.5B retrainings, failure disclosure has 3.9× larger cross-run SD than solve rate. (B) From one mid-RL state at step 150, ten continuations differing only in future rollout randomness end with failure-disclosure rates from 0.11 to 0.88. This is the widest of eight mid-RL branch-and-continue replication tests; across its eight ten-continuation cells, the median within-cell rate SD is 0.10 (Appendix Table 19). (C) In 20 matched Qwen2.5-1.5B pairs, a reference-policy penalty applied to failed, well-formed responses substantially reduces the run-to-run spread of disclosure. The unanchored points in C use the same underlying checkpoints as A, but an independent probe sampling draw used for the anchoring comparison; the measured rates are therefore similar but not identical. Across the distribution figures, dots show retraining runs, short horizontal lines show means, and faint vertical lines show observed min–max ranges.

capability being optimized. Across 20 independent runs of the same Qwen2.5-1.5B training procedure, disclosure varies 3.9× more than task solve rate. The spread remains on a broader, unselected evaluation set, so it is not created by the failure-heavy probe. We reproduce this selectivity in a second set of Qwen-1.5B runs, in OLMo-2-1B, and on a second reasoning task. Additional experiments test its boundaries under stabilized PPO and at 7B. A separate instruction-conditioned 32B experiment asks whether the effect depends on report SFT: reporting is elicited through the prompt instead. We treat both larger-model experiments as boundary evidence since capability also varies more there. The main point is not that every model shows the same instability, but that one behavior from a training run can be far less consistent than its headline capability.

We next ask why runs following the same training procedure can end up with such different reporting behavior. The answer is not simply "different seeds." With the seed and data schedule fixed, changing only low-level floating-point execution details can redirect the learned reporting behavior. Restoring the *same full mid-RL training state* at step 150 and changing only future rollout randomness can also produce heavily differing reporting behavior by step 300, showing that the divergence can arise later in training. Further, we treat failure disclosure as a sequence of linked decisions: checking the attempted answer, reaching and entering a reporting route, and completing an admission. Our interventions separate these stages in different settings. In the primary Countdown format, a learned paragraph transition affects later reporting; in grid and inline-report settings, the boundary appears elsewhere. The paragraph transition is one format-specific bottleneck in a broader multi-stage process, not a universal marker of whether the model recognized its failure. Experiments with other behaviors show a similar pattern. A weakly constrained neutral phrase becomes highly variable across runs, while the same phrase is almost perfectly reproducible when tied to an explicit deterministic cue, demonstrating that this sensitivity is not unique to failure disclosure. More broadly, outcome RL can leave auxiliary behaviors such as failure disclosure weakly constrained, allowing small floating-point or stochastic differences to grow through later on-policy training.

Finally, we ask whether this training-path sensitivity can be reduced. We add a reference-policy penalty only to failed, well-formed responses, which we call *failure-conditional reference anchoring*. This substantially reduces disclosure variability across repeated training runs on two verifiable reasoning tasks. This reduces task performance in the evaluated 1.5B settings, while the 19 tested 7B runs show a small solve gain. After an anchored training history, keeping the anchor on also reduces

how much future continuations from the same mid-RL state diverge in reporting behavior. Here, reference anchoring shows that a measured reproducibility failure can be reduced and reveals the setting-dependent tradeoff between preserving reporting behavior and learning the rewarded task.

Overall, we find that safety-relevant behavior can be far less consistent across RL training runs than task capability. This variability can arise from the same training state and can be reduced with reference anchoring. The broader lesson is that post-training reproducibility should be evaluated not only for task performance but also for auxiliary behaviors that matter for oversight.

2 EXPERIMENTAL FRAMEWORK

Settings. Our main setting is **Countdown**, an arithmetic task used in recent reasoning-RL work (Pan et al., 2025; Gandhi et al., 2025). Given four numbers and a target, the model must build an arithmetic expression that uses each number exactly once and reaches the target. Because the result can be checked exactly, success and failure are objectively defined. We also use **shortest path** from Reasoning Gym (Stojanovski et al., 2025) as a structurally different verifiable reasoning task. Qwen2.5-1.5B is our main model, while OLMo-2-1B (Team OLMo et al., 2025) provides a second model family. Qwen-7B tests persistence at larger scale, while Qwen2.5-32B-Instruct (Qwen et al., 2024) provides a separate instruction-conditioned boundary that does not use report SFT. Our main experiments use GRPO, with stabilized PPO (Schulman et al., 2017) as an algorithmic boundary check. We also use grid/constraint and MBPP-style code tasks (Austin et al., 2021) for specific checking, routing, and intervention-boundary experiments (Appendix A).

Post-training process. Most experiments begin from an SFT warm start already exhibiting failure reporting, followed by KL-free outcome-only GRPO (Shao et al., 2024; Liu et al., 2025). The reward scores task success but does *not* directly reward disclosure, and our primary runs use no reference-policy KL. This matches modern KL-free reasoning-RL recipes such as Dr. GRPO and DAPO (Liu et al., 2025; Yu et al., 2025) and lets us study what happens to a behavior that the task reward does not directly affect. We repeat this post-training process while controlling or changing the random seed, low-level execution configuration, or future rollout randomness. The 32B experiment instead starts directly from Qwen2.5-32B-Instruct and elicits failure reporting through the prompt, providing a test of whether the phenomenon depends on installing reporting through SFT (Appendix A).

Failure disclosure. Among terminated task failures, we measure whether the model acknowledges that its attempted solution failed, rather than ending without an acknowledgment or falsely presenting the attempt as successful. Terminated format-invalid failures count as non-disclosures, while unterminated completions are excluded. We use a conservative detector in the main text; broader semantic checks, blind audits, and qualitative output examples are reported in Appendix B.

Three sources of retraining variation. A *retraining seed* gives an independent draw by changing the shuffled prompt order and the Python/NumPy/Torch sampling streams. An *execution configuration* changes low-level floating-point computation while keeping the seed and data schedule fixed, for example by changing the micro-batch/gradient-accumulation split or GPU/hardware setup. A *continuation key* is more controlled: We restore the exact same saved model, optimizer, scheduler, trainer, and RNG state, then change only the future rollout while keeping the parent seed and data order fixed. These manipulations let us move from ordinary retraining variability toward more direct tests of how controlled training differences causally redirect behavior.

Statistics. Our main quantities are task solve rate, failure-disclosure rate conditional on failure, and variation across retraining runs. We compare variability using standard-deviation ratios, with log ratios used for statistical inference. Hierarchical $\hat{\tau}$ estimates and other robustness measures are reported where useful. The retraining run is the main unit of analysis. Appendices A.5 and B provide detector definitions, estimands, confidence intervals, and robustness analyses.

3 CAPABILITY CAN REPRODUCE WHILE FAILURE DISCLOSURE DOES NOT

Primary retrainings. We run 20 independent Qwen2.5-1.5B Countdown retrainings for 300 steps using the same fixed execution configuration. As Figure 2A shows, failure disclosure varies widely across runs, from 0.257 to 0.903, while solve stays much narrower at 0.340–0.490. Disclosure is about **3.9**$\times$ more variable than solve, and an independent set of 20 Qwen-1.5B retrainings from a

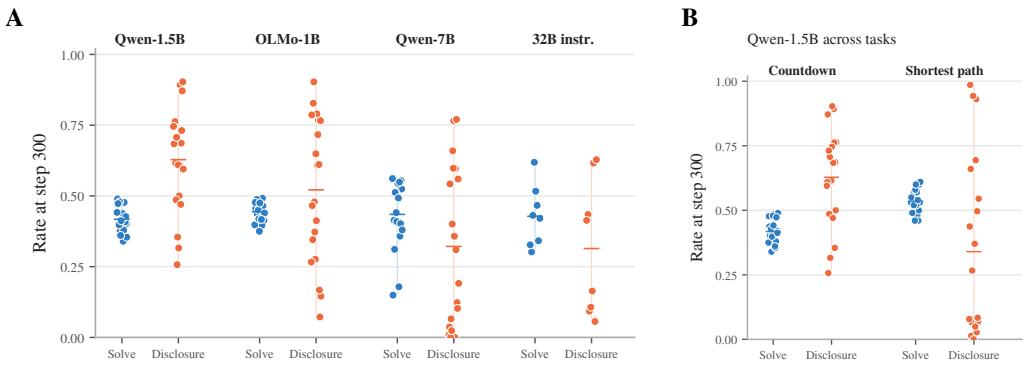

Figure 2: **Failure-disclosure variability appears across models, tasks, and reporting setups.** (A) On Countdown, failure disclosure varies substantially more across retrainings than solve for Qwen2.5-1.5B & OLMo-2-1B. Qwen-7B is larger-scale evidence; instruction-conditioned Qwen2.5-32B tests reporting without an SFT warm start; capability is also broader. (B) This pattern also appears across tasks with Qwen2.5-1.5B: Failure disclosure varies substantially more than task performance on both Countdown and shortest path, using each task's primary capability evaluation.

separate SFT reference shows the same selective reproducibility. Broader detectors and the full, unselected dev set give the same qualitative conclusion (Appendices C.1 and B).

Across model families and tasks. In Figure 2A, across 20 OLMo-2-1B Countdown retrainings, disclosure ranges from 0.072 to 0.903, while solve stays much more stable at 0.375–0.492; disclosure is about 7.5× more variable than solve. On shortest-path (Figure 2B), 20 Qwen-1.5B retrainings also cover nearly the full disclosure range by step 300, while greedy solve on the 100-problem dev set spans 0.46–0.61. Full estimates, including the failure-heavy probe surface, are in Appendix C.

Across scales and reporting setups. Across 19 Qwen-7B retrainings using the same execution configuration, disclosure remains highly variable, but solve also varies more. We use 7B as evidence that the instability does not disappear at larger scale. The 32B experiment addresses a different concern: whether the effect depends on failure reporting first being installed through SFT. Starting directly from Qwen2.5-32B-Instruct, we elicit reporting through the prompt. Across eight retrainings, reporting also varies substantially: the frozen detector gives a conservative range of 0.056–0.628 at step 300, while blind-read calibration gives 0.173–1.000; solve spans 0.302–0.619. We therefore treat 32B as an instruction-conditioned boundary test (Appendices B and C).

When models do not disclose a failure, they either end without acknowledging it or falsely present a wrong answer as correct. False success claims range from 9–28% of terminated failures across the 20 unanchored 1.5B runs and from 2–59% across the 19 unanchored 7B runs (Appendix Table 8).

Algorithm and horizon boundaries. Eight stabilized-PPO runs provide a test of algorithm generalization: we also see the reporting variability, but a smaller absolute spread than GRPO's (Appendix C). Further, disclosure does not decline at the same rate in every run. On the primary probe, the step-300 mean is 0.628 vs. 0.677 for the reference model, with 10 runs above the reference and 10 below. The ordering of runs is weakly preserved over training, and the longer-horizon experiments are mixed. We thus treat step 300 as a fixed comparison point, not a convergence point. Broader detectors, failure-composition reweighting, evaluation-noise adjustment, blind audits, alternate dispersion measures, horizon comparisons, and smaller recipe variants do not indicate that this phenomenon is a simple measurement or recipe artifact (Appendices B–J, Table 37). The distinction from generic seed sensitivity (Henderson et al., 2018; Fehlauer et al., 2025; Bui et al., 2025) is *selective reproducibility*: Behaviors from the same post-training procedure can vary quite differently.

4 How behavioral divergence arises

Seeds change many aspects of training at once. To better understand how behavioral divergence arises, we isolate specific sources of divergence, shown in Figure 3: (A) floating-point execution

differences can redirect training, (B) reporting can still diverge later in RL, (C/D) at which reporting stage runs differ, and (E) whether this extends beyond failure semantics.

4.1 Small floating-point and stochastic differences can redirect training

Execution configuration. Figure 3A changes only the micro-batch $\times$ gradient-accumulation split at fixed seed and global batch. Across six execution configurations, disclosure ranges from 0.186 to 0.848 while solve stays at 0.327–0.463. With the Dr. GRPO loss, equal-sized micro-batches, split-invariant accumulation scaling, and dropout inactive, the objective and exact-arithmetic gradient remain identical; only floating-point kernel shapes and reduction order differ (Appendix D). As a control, repeating the same execution configuration reproduces the result bit-for-bit in 5/5 runs. Changing hardware (GPU), e.g., H200 vs. B200, can change generation from the very first training step, so we treat this as another source of floating-point execution differences. Overall, very small floating-point differences can grow into large behavioral differences during outcome-RL post-training, consistent with broader observations about numerical sensitivity in neural-network training (Altıntaş et al., 2025; Yuan et al., 2025; Shanmugavelu et al., 2024).

Continuations from the mid-RL state. Figure 3B asks whether reporting can still diverge later in RL training. We restore the same model, optimizer, scheduler, trainer, and RNG state at step 150, keep the data order fixed, and change only future rollout randomness. In the widest of eight mid-RL continuation tests, ten continuations end with disclosure from 0.112 to 0.880 at step 300 while solve varies much less; the median within-cell rate SD across all eight tests is 0.10. Thus the reporting differences need not be fixed at initialization or early in RL training (Appendix D).

4.2 Failure disclosure is a multi-stage behavior

Failure disclosure is not a single decision. After an unsuccessful attempt, a model may need to check its answer, use evidence that it failed, reach a point where it can report the failure, begin the report, and complete the admission. Disclosure can break at any stage. In the grid task, adding checking examples in the post-training mix produces grounded checks and failure verdicts that are otherwise largely absent (Appendix E). In a codeword control, preserving a reporting-related phrase does not ensure that the model continues to use it when failing (Appendix F). We examine the later reporting stages in Countdown. Figure 3C separates reaching a point where a report could begin from admitting failure afterward. Across 40 checkpoints and report wording combinations, models usually reach this point (median probability 0.952), but admission rates afterward vary widely. On this evaluation surface, observed differences occur after the model has an opportunity to report. This does not tell us if a silent model failed to recognize its error or recognized it but did not express it.

Figure 3D intervenes at the same point. Restarting generation without inserting any characters changes the spread slightly. Inserting the learned two-newline paragraph transition makes later admission more frequent and consistent. Raw spread narrows in all six comparisons, and the reduction remains after accounting for the higher admission rate in the primary recipe and in one related comparison. Adding a bracketed message stating that the verifier found the previous attempt incorrect does not further make reporting more consistent, while supplying a short report prefix has a larger effect. In the primary response format, access to the later reporting behavior is thus sensitive to the transition to the report and cues that initiate it; failure evidence alone is less effective. We treat this paragraph transition as a learned cue here rather than a universal marker of failure recognition. The inline-report 7B and single-line grid settings place the weakest transition elsewhere. Overall, failure disclosure has separable stages, but which stage is fragile depends on the task and response format.

4.3 Neutral control phrases show that weakly constrained behaviors are especially susceptible

To test whether the instability is specific to failure semantics, we use the fixed neutral phrase *"Thanks for the question."* In the weakly constrained condition, the phrase appears in training but no prompt feature determines when it should be used. Its frequency later ranges from 0 to 0.90 across retrainings. We then add a simple outcome-independent rule: Each prompt ends with either *Marker: A* or *Marker: B*, and the model is trained to begin with *"Thanks for the question."* for A but omit it for B. Under this rule, all 20 runs follow the instruction perfectly at step 150 and 19 of 20 do so at step 300.

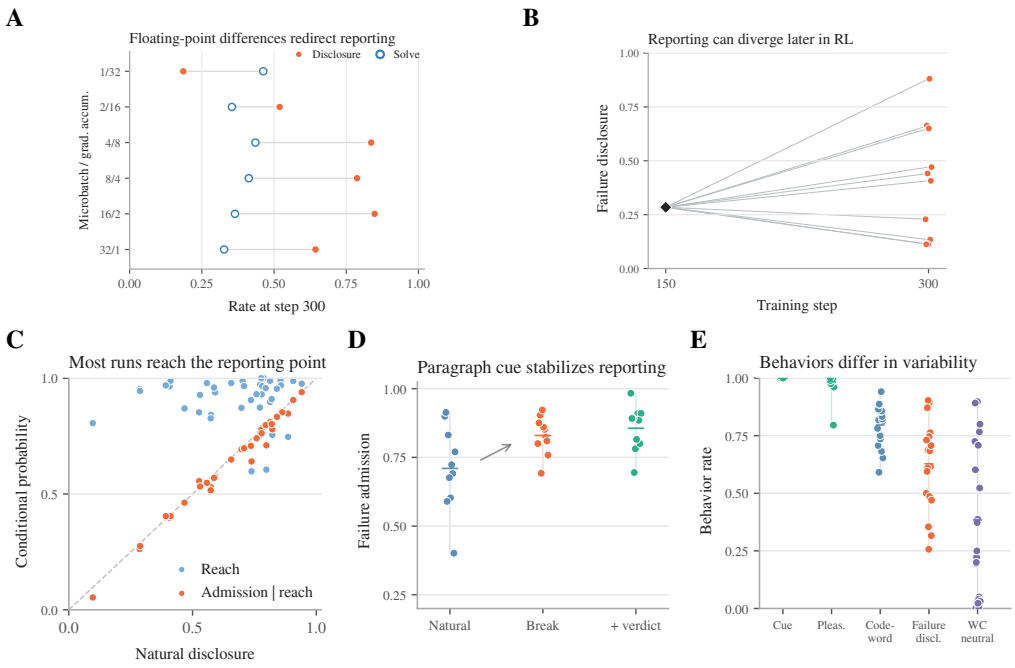

Figure 3: **Training differences and behavioral cues shape reporting variability.** (A) Each row uses a different execution configuration, with solve and disclosure measured from the same run. Changing only the configuration moves failure disclosure from 0.186 to 0.848, while solve remains 0.327–0.463. (B) Starting from the same mid-RL training state, changing only future rollout randomness produces widely different disclosure outcomes. (C) Models nearly always reach a point where a report could begin, but differ in whether they admit failure after. (D) Inserting the paragraph break from the learned report template makes later admission more frequent and consistent in the primary recipe; adding a verifier verdict does not help further. (E) The same neutral phrase varies widely when weakly constrained but is nearly reproducible when tied to an explicit cue. The primary disclosure spread, a failure-conditioned neutral codeword, and success-side pleasantries are shown for comparison. Appendices E and F report the additional localization and control results.

Other controls show the same pattern: Success-side pleasantries (a family of polite closing sentences associated with solved SFT examples) and a failure-conditioned neutral codeword/phrase *"A copper kettle rests on the kitchen shelf."* are also relatively stable (Appendix F). Figure 3E places failure disclosure between these more strongly constrained controls and the more weakly constrained neutral phrase. Additional formulation variants are reported in Appendix F. These controls show that high variability is neither unique to failure reporting nor a property of a particular string. What matters is how strongly training specifies when the behavior should occur: Weakly specified behaviors can drift across runs, while an explicit rule can make the same behavior highly reproducible.

4.4 Collateral credit can affect reporting, but does not explain the spread

With outcome-level rewards, tokens can receive credit or blame even when they did not directly cause the final outcome. In our mixed groups, this credit is by default adverse to failure reporting, and all 517 relevant groups have that sign. In a three-pair pilot experiment, protecting the reporting span from this signal raises disclosure by 0.19 on average, showing that local credit can causally affect how often the model reports failure. However, follow-up experiments with up to 20 pairs do not demonstrate that protecting the report span reliably reduces variation across retrainings. Further, removing the adverse failure credit entirely does not eliminate the run-to-run spread and lowers task performance. Collateral credit can thus affect reporting, but the current evidence does not establish it as a general explanation for the level or variability. Appendix G reports the full results.

Overall, Figure 3 shows that small floating-point/stochastic differences can push training toward different weakly constrained behaviors, and the differences can grow through the model's later training.

5 REFERENCE ANCHORING STABILIZES FAILURE DISCLOSURE

The most direct way to preserve failure reporting is to reward it, and prior work shows that such rewards can be effective (Joglekar et al., 2025). This is attractive when the desired behavior is known in advance, can be scored reliably, and can be added to the objective without creating unwanted incentives. Many useful auxiliary behaviors, however, are difficult to enumerate or score directly. Our main question is hence different: Can an existing reporting behavior remain reproducible when the task reward does not reward it? Our code experiments show that direct reward can increase or recover reporting while the reporting action remains in sampling support, but cannot reinforce an action that has disappeared from sampled behavior (Appendix G, Table 26).

Reference anchoring instead tries to preserve a broader part of the starting policy by adding a penalty that discourages the model from drifting too far without requiring a separate reward definition for every behavior. Our failure-conditional version applies the penalty to every completion token in incorrect, format-valid rollouts, and no penalty to solved or format-invalid rollouts. We fixed $\beta = 0.04$ before reading the outcomes and compare it with $\beta = 0$ under the same trainer. For 1.5B Countdown, the $\beta = 0$ comparator is the existing set of 20 outcome-only runs rather than a separate control set; Appendix H reports checks that the implementations agree, along with a small coefficient sweep (we find clear narrowing at $\beta = 0.02$ but do not establish it at $\beta = 0.01$; we treat these settings as a limited robustness check). Unlike prior uses of reference regularization to limit behavioral drift (Elcock et al., 2026) or toward a common prior to reduce policy differences across continuous-control runs (Hussing et al., 2026), we ask whether a failure-conditional anchor makes a specific reporting behavior more reproducible across language-model retrainings and stochastic continuations.

Anchoring narrows retraining variability. Figure 4A summarizes the main intervention results. Comparing 20 anchored Qwen-1.5B Countdown runs with their matched outcome-only runs, failure-conditional anchoring reduces disclosure SD from 0.189 to 0.089 and solve SD from 0.048 to 0.029: Both readouts narrow, but disclosure substantially more. All-token anchoring produces comparable narrowing, while a direct matched-seed comparison shows that failure-conditional anchoring preserves 2.4 percentage points more solve accuracy. We thus focus on failure-conditional anchoring because it covers fewer tokens and has a smaller capability cost. The two anchorings use the same per-token coefficient but are not matched for total regularization, so this comparison does not prove that one is more efficient at an equal budget. On shortest-path, disclosure variability falls to $\sim$30% of the unanchored level while solve decreases by $\sim$0.08. Full intervals and analyses are in Appendix H. In both 1.5B settings, reference anchoring narrows disclosure variability at a cost of some task performance. A post-hoc comparison at matched mean solve rate shows that slower progress explains part of the Countdown result. Disclosure SD is 0.139 at the matched unanchored checkpoint, compared with 0.189 at unanchored step 300 and 0.089 under anchoring. The anchored value is 36% lower than the progress-matched value. Thus anchoring still narrows disclosure beyond matched capability progress, but this comparison does not fully separate the two effects (Appendix H).

At 7B, the trade-off looks different. The initial comparison included 11 matched pairs, with one anchored and one unanchored run in each pair. After seeing that result, we prospectively froze the remaining eight eligible pairs and repeated the experiment with unchanged training and evaluation settings. The eight new pairs showed the same narrowing. Across all 19 pairs, anchoring reduces disclosure SD from 0.28 to 0.09 (32% of the unanchored value), and the conclusion is unchanged when any pair is removed. See Appendix H.4 for details. This narrowing remains after accounting for the different mean disclosure rates: On a latent logit scale, the anchored 7B runs have about one quarter of the between-run dispersion of the unanchored runs (Appendix Table 28). At 7B, anchoring also lowers both forms of nondisclosure: False success claims fall from 0.27 to 0.14 of terminated failures, while silent closes fall from 0.21 to 0.02. False success claims therefore make up most of the nondisclosure that remains under anchoring (Appendix Table 8). The combined analysis also shows a small 6.9-point gain in probe solve rate. Because this is a single model–task setting, we do not interpret the solve gain as evidence that anchoring becomes cost-free at larger scales.

Across all three settings, the consistent effect of anchoring is lower run-to-run variability. The effect on mean disclosure depends on the setting: It stays roughly unchanged in 1.5B Countdown, is

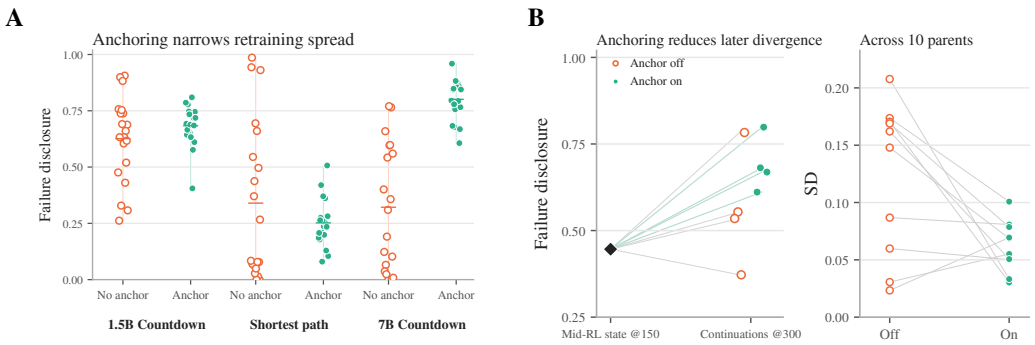

Figure 4: **Reference anchoring reduces retraining variability and future divergence from the same mid-RL state.** (A) Anchoring makes disclosure more consistent across retrainings in 1.5B and 7B Countdown and shortest-path. In 1.5B Countdown, disclosure SD falls from 0.189 to 0.089; at matched mean solve, the unanchored SD is 0.139. At 7B, disclosure SD falls from 0.281 to 0.090, and probe solve is 0.069 higher under anchoring. Changes in average disclosure differ across settings. (B) After training with anchoring, keeping the anchor on leads to less variation across future continuations. The left plot shows four anchor-off and four anchor-on continuations from one saved mid-RL state. The right plot summarizes all ten mid-RL states. Token-level localization results are described in the text and Appendix I; the 7B sensitivity analysis is in Appendix H.4.

somewhat lower (not established) on shortest-path, and substantially higher at 7B. We should view reference anchoring as a way to stabilize an existing behavior, rather than as a direct reward for it.

Figure 4B asks whether anchoring also reduces later-RL divergence from a shared mid-RL state. Across ten anchored step-150 parent states, matched anchor-on vs. anchor-off continuations give a within-state disclosure-SD ratio of 0.47, with lower variability under anchoring in 8/10 parents. Full design and capability effects are in Appendix I.

Where does the stabilization act? We next ask which part of a failed trajectory needs to be anchored. Applying the reference penalty only to a short tail after the attempt, or to a 16-token region around the reporting point, does not detectably reduce variability. Because these regions cover only a small fraction of the trajectory, these null results do not tell us whether the locations are unimportant or whether the intervention there is simply too weak. In contrast, anchoring only the failed-reasoning portion roughly halves disclosure variability, while solve decreases $\sim$0.06, similar to the full failure-conditional anchor. Further gradient and policy-drift analyses in Appendix I support this. The reference gradient from the failed-reasoning portion is at least 94% as large as the full-anchor gradient and points in nearly the same direction. Most of the measured drift from the reference policy also occurs before the reporting point, while the tail and seam contribute much smaller gradients. These results do not support a simple picture in which anchoring works only by protecting the final report tokens. At the strengths we tested, most of the stabilizing effect comes from constraining the broader failed-reasoning trajectory, which is also the part of the response most closely tied to task learning. This offers a plausible explanation for why stabilization and capability cost can appear together, but it is not a complete account of the underlying dynamics (Appendix I).

6 Related Work

Failure awareness, verification, and reporting. Knowing, finding, admitting, and correcting an error are different LLM abilities (Kadavath et al., 2022; Tyen et al., 2024; Yang & Jia, 2026; Chen et al., 2026). Related work studies self-evaluation, confidence, retraction, and correction (Mavi et al., 2025; Zhao et al., 2026; Xie et al., 2026; Tsui, 2026; Huang et al., 2024). We ask a separate behavioral question: after a failed attempt, does the model tell the user, and is this behavior stable?

Incentives, abstention, and explicit supervision. Post-training can change how models communicate mistakes (Wen et al., 2025; Chandna et al., 2026). Accuracy rewards can favor guessing over abstention, while direct supervision can train confession or abstention (Kalai et al., 2026; Joglekar

et al., 2025; Zhai et al., 2026). We study an unrewarded behavior and compare direct reward with preservation. Process supervision is related to our collateral-credit tests (Liu et al., 2026c).

Underspecification, reproducibility, and preservation. Similar task performance can hide behavioral differences (D'Amour et al., 2022). Seed sensitivity is known, and small floating-point differences can redirect training (Henderson et al., 2018; Fehlauer et al., 2025; Bui et al., 2025; Altıntaş et al., 2025; Yuan et al., 2025; Shanmugavelu et al., 2024). Recent work studies safety-relevant drift and reference-based preservation (Schreiber & Goldstein, 2026; Betley et al., 2025; Elcock et al., 2026; Soligo et al., 2026). We connect these through selective behavioral reproducibility, mid-RL branching, and reference anchoring. See Appendix K for extended related work.

7 Discussion

We would like to train models that reliably produce safety-relevant behaviors. To assess progress towards this goal, we measure cross-run variability in whether models admit failure. While outcome-based RL runs repeatedly reach similar task capability, they produce very different failure-reporting behavior. Thus, capability measurements should be supplemented with those of safety-relevant behaviors, and these behaviors should be considered when designing training objectives.

Safety-relevant behaviors are part of a broader class of behaviors that can vary across retrainings because they are not explicitly included in the training objective. The consequences depend on the behavior: Variation in a pleasantry is mostly cosmetic, while failure disclosure changes what information reaches an overseer. This variability is also not inevitable: Smaller boundary experiments with other model families such as Gemma and Phi do not show the same large spread (Appendix C, Table 16). A useful question is thus which valued behaviors remain weakly constrained by a particular post-training objective, and which model families are robust vs. susceptible to this instability.

The causal experiments show how this can arise. Repeating the same execution configuration reproduces the result, while changing it or future rollout randomness can redirect reporting. Continuations from the same mid-RL state show that reporting can still diverge later in training rather than being fixed early or at initialization. The multi-stage experiments show why failure disclosure should not be treated as one indivisible behavior. Checking an answer, entering a reporting route, and final admission can separate, and the bottleneck changes with the task and response format.

Direct reward is suitable when the desired behavior can be defined and scored, but cannot reinforce actions that have disappeared from sampling. Reference anchoring instead tries to preserve an existing policy. Across experiments, anchoring reduces disclosure variability, including after an anchored training history. Its effects on mean disclosure and capability differ by setting. Behavioral preservation is not a free regularizer but an empirical tradeoff to evaluate jointly with task learning.

Future directions. Predict which behaviors will be weakly constrained before training; test reproducibility on more naturalistic tasks, broader post-training pipelines, and larger models; develop preservation methods that interfere less with task learning. These connect to the goal of preserving monitorability-relevant behavior during optimization (Guan et al., 2026; Emmons et al., 2025).

Limitations. By using controlled tasks, we can objectively determine whether an answer failed, but we cannot tell how common the same effect is in deployed systems. Our evaluations use fixed prompt sets rather than representative deployment samples. The 7B anchoring evidence comes from one model–task setting, so we claim no scaling law. Our localization intervention is specific to the tested response format and does not reveal the model's internal recognition process. Finally, our boundary experiments show that large disclosure variation is not universal. We treat it as an important failure mode to measure and understand, not a guaranteed property of outcome-based RL.

The practical takeaway: If a post-training pipeline is expected to preserve a safety-relevant behavior, it should be measured across multiple retrainings and evaluated beside task capability. Reproducing the reward or accuracy alone can hide large differences in what the model communicates when it fails. Post-training is expensive, so it is tempting to compare recipes using a single run or metric. We show this can be misleading: A pipeline can reliably reproduce what it rewards while leaving a valued but weakly constrained behavior dependent on the specific training trajectory. When a behavior matters for oversight, it should become an explicit part of the evaluation and training objective.

AI USE STATEMENT

We used generative AI to assist with experimental planning and design, implementation, analysis and result interpretation, literature search, and manuscript drafting and editing. We reviewed and verified AI-assisted code, results, citations, claims, and prose against primary sources and experimental artifacts. The authors directed the research and take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We study a controlled reporting behavior in language models. We involve no human subjects nor establish deployment harm or prevalence. Our goal is to improve evaluation & preservation of safety-relevant behavior during post-training. We report negative & boundary results that limit claims.

REPRODUCIBILITY STATEMENT

Appendix A documents models, tasks, training, seeds, and estimands; later appendices give detector audits, summaries across runs, and robustness analyses. We will release all of the code and data.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. Phi-4 technical report. arXiv preprint arXiv:2412.08905, 2024. URL https://arxiv.org/abs/2412.08905.

Gül Sena Altıntaş, Devin Kwok, Colin Raffel, and David Rolnick. The butterfly effect: Neural network training trajectories are highly sensitive to initial conditions. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 1314–1342. PMLR, 2025. URL https://proceedings.mlr.press/v267/altintas25a.html.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. arXiv preprint arXiv:2108.07732, 2021. URL https://arxiv.org/abs/2108.07732.

Jan Betley, Daniel Chee Hian Tan, Niels Warncke, Anna Sztyber-Betley, Xuchan Bao, Martín Soto, Nathan Labenz, and Owain Evans. Emergent misalignment: Narrow finetuning can produce broadly misaligned LLMs. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 4043–4068. PMLR, 2025. URL https://proceedings.mlr.press/v267/betley25a.html.

Jan Betley, Niels Warncke, Anna Sztyber-Betley, Daniel Tan, Xuchan Bao, Martín Soto, Megha Srivastava, Nathan Labenz, and Owain Evans. Training large language models on narrow tasks can lead to broad misalignment. *Nature*, 649(8097):584–589, 2026. doi: 10.1038/s41586-025-09937-5. URL https://doi.org/10.1038/s41586-025-09937-5.

Nghia Tuan Bui, Guergana K Savova, and Lijing Wang. Assessing the macro and micro effects of random seeds on fine-tuning large language models. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Short Papers)*, pp. 41–46, Mumbai, India, 2025. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics. doi: 10.18653/v1/2025.ijcnlp-short.3. URL https://aclanthology.org/2025.ijcnlp-short.3/.

Aaryan Chandna, Lukas Fluri, and Micah Carroll. Is the evidence in 'Language Models Learn to Mislead Humans via RLHF' valid? ICLR 2026 Blogposts Track, April 2026. URL https://iclr-blogposts.github.io/2026/blog/2026/mislead-lm/.

Yuxin Chen, Yu Wang, Yi Zhang, Ziang Ye, Zhengzhou Cai, Yaorui Shi, Qi Gu, Hui Su, Xunliang Cai, Xiang Wang, An Zhang, and Tat-Seng Chua. Learning to self-verify makes language models better reasoners. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=rSDj9EM1HV.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, Farhad Hormozdiari, Neil Houlsby, Shaobo Hou, Ghassen Jerfel, Alan Karthikesalingam, Mario Lucic, Yian Ma, Cory McLean, Diana Mincu, Akinori Mitani, Andrea Montanari, Zachary Nado, Vivek Natarajan, Christopher Nielson, Thomas F. Osborne, Rajiv Raman, Kim Ramasamy, Rory Sayres, Jessica Schrouff, Martin Seneviratne, Shannon Sequeira, Harini Suresh, Victor Veitch, Max Vladymyrov, Xuezhi Wang, Kellie Webster, Steve Yadlowsky, Taedong Yun, Xiaohua Zhai, and D. Sculley. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*, 23(226):1–61, 2022. URL https://www.jmlr.org/papers/v23/20-1335.html.

Eric Eaton, Marcel Hussing, Michael Kearns, Aaron Roth, Sikata Bela Sengupta, and Jessica Sorrell. Replicable reinforcement learning with linear function approximation. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=Ccflf8sqjF.

James Elcock, William F. Shen, Xinchi Qiu, and Nicholas D. Lane. How LLM task-adaptation reshapes alignment: A multi-dimensional study of behavioral and representational drift. arXiv preprint arXiv:2607.22676, 2026. URL https://arxiv.org/abs/2607.22676.

Scott Emmons, Erik Jenner, David K. Elson, Rif A. Saurous, Senthooran Rajamanoharan, Heng Chen, Irhum Shafkat, and Rohin Shah. When chain of thought is necessary, language models struggle to evade monitors. arXiv preprint arXiv:2507.05246, 2025. URL https://arxiv.org/abs/2507.05246.

Finlay Fehlauer, Kyle Mahowald, and Tiago Pimentel. Convergence and divergence of language models under different random seeds. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 32982–32991, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1675. URL https://aclanthology.org/2025.emnlp-main.1675/.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STaRs. In *Proceedings of the Conference on Language Modeling (COLM)*, 2025. URL https://openreview.net/forum?id=QGJ9ttXLTy.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, et al. Gemma 2: Improving open language models at a practical size. arXiv preprint arXiv:2408.00118, 2024. URL https://arxiv.org/abs/2408.00118.

Satvik Golechha, Sid Black, and Joseph Bloom. Preliminary investigation: KL penalties in RL can increase CoT unfaithfulness. Research post, LessWrong, June 2026. URL https://www.lesswrong.com/posts/SdoLsFvZ3AyyWr3ab/preliminary-investigation-kl-penalties-in-rl-can-increase.

Raja Gond, Aditya K Kamath, Ramachandran Ramjee, and Ashish Panwar. LLM-42: Enabling determinism in LLM inference with verified speculation. In *Proceedings of the 32nd ACM Symposium on Operating Systems Principles (SOSP)*, 2026. URL https://arxiv.org/abs/2601.17768.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. arXiv preprint arXiv:2407.21783, 2024. URL https://arxiv.org/abs/2407.21783.

Melody Y. Guan, Miles Wang, Micah Carroll, Zehao Dou, Annie Y. Wei, Marcus Williams, Benjamin Arnav, Joost Huizinga, Ian Kivlichan, Mia Glaese, Jakub Pachocki, and Bowen Baker. Monitoring monitorability. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=b82fgbMVpz.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Horace He and Thinking Machines Lab. Defeating nondeterminism in LLM inference. Thinking Machines Lab: Connectionism (research post), September 2025. URL https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/.

Lukas Helff, Quentin Delfosse, David Steinmann, Ruben Härle, Hikaru Shindo, Patrick Schramowski, Wolfgang Stammer, Kristian Kersting, and Felix Friedrich. LLMs gaming verifiers: RLVR can lead to reward hacking. In *ICLR 2026 Workshop on Logical Reasoning of Large Language Models*, 2026. URL https://openreview.net/forum?id=4B3WfRNqe3.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, volume 32, pp. 3207–3214, 2018. doi: 10.1609/aaai.v32i1.11694. URL https://ojs.aaai.org/index.php/AAAI/article/view/11694.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Marcel Hussing, Liv G. d'Aliberti, Claas Voelcker, Benjamin Eysenbach, and Eric Eaton. Behavior-consistent deep reinforcement learning. arXiv preprint arXiv:2605.21214, 2026. URL https://arxiv.org/abs/2605.21214.

Manas Joglekar, Jeremy Chen, Gabriel Wu, Jason Yosinski, Jasmine Wang, Boaz Barak, and Amelia Glaese. Training LLMs for honesty via confessions. arXiv preprint arXiv:2512.08093, 2025. URL https://arxiv.org/abs/2512.08093.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. arXiv preprint arXiv:2207.05221, 2022. URL https://arxiv.org/abs/2207.05221.

Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. Evaluating large language models for accuracy incentivizes hallucinations. *Nature*, 653(8116):1047–1051, 2026. doi: 10.1038/s41586-026-10549-w. URL https://doi.org/10.1038/s41586-026-10549-w.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can LLMs actually correct their own mistakes? A critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024. doi: 10.1162/tacl_a_00713. URL https://aclanthology.org/2024.tacl-1.78/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP '23)*, pp. 611–626, Koblenz, Germany, 2023. ACM. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Emmy Liu, Varun Gangal, Michael Yu, Zhuofu Tao, Karan Singh, Sachin Kumar, and Steven Y. Feng. HalluWorld: A controlled benchmark for hallucination via reference world models. arXiv preprint arXiv:2605.19341, 2026a. URL https://arxiv.org/abs/2605.19341.

Emmy Liu, Varun Prashant Gangal, Chelsea Zou, Michael Yu, Xiaoqi Huang, Alex Chang, Zhuofu Tao, Karanpartap Singh, Sachin Kumar, and Steven Y. Feng. Position: We need a unified definition of hallucination (It's the world model, stupid!). In *Proceedings of the 43rd International Conference on Machine Learning (ICML), Position Paper Track*, 2026b. URL https://openreview.net/forum?id=t4lDjFq5lb.

Haolin Liu, Dian Yu, Sidi Lu, Yujun Zhou, Rui Liu, Zhenwen Liang, Haitao Mi, Chen-Yu Wei, and Dong Yu. Save the good prefix: Precise error penalization via process-supervised RL to enhance LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 35450–35477. Association for Computational Linguistics, 2026c. doi: 10.18653/v1/2026.findings-acl.1767. URL https://aclanthology.org/2026.findings-acl.1767/.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Proceedings of the Conference on Language Modeling (COLM)*, 2025. URL https://openreview.net/forum?id=5PAF7PAY2Y.

Matt MacDermott, Qiyao Wei, Rada Djoneva, and Francis Rhys Ward. Reasoning under pressure: How do training incentives influence chain-of-thought monitorability? arXiv preprint arXiv:2512.00218, 2025. URL https://arxiv.org/abs/2512.00218.

Vaibhav Mavi, Shubh Jaroria, and Weiqi Sun. Self-evaluating LLMs for multi-step tasks: Stepwise confidence estimation for failure detection. In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*, 2025. URL https://arxiv.org/abs/2511.07364.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. TinyZero. GitHub repository, 2025. URL https://github.com/Jiayi-Pan/TinyZero.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115, 2024. URL https://arxiv.org/abs/2412.15115.

Joel Schreiber and Ariel Goldstein. Overtrained, not misaligned. arXiv preprint arXiv:2605.12199, 2026. URL https://arxiv.org/abs/2605.12199.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017. URL https://arxiv.org/abs/1707.06347.

Sanjif Shanmugavelu, Mathieu Taillefumier, Christopher Culver, Oscar Hernandez, Mark Coletti, and Ada Sedova. Impacts of floating-point non-associativity on reproducibility for HPC and deep learning applications. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 170–179, Atlanta, GA, USA, 2024. IEEE. doi: 10.1109/SCW63240.2024.00028. URL https://doi.org/10.1109/SCW63240.2024.00028.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024. URL https://arxiv.org/abs/2402.03300.

Anna Soligo, Edward Turner, Senthooran Rajamanoharan, and Neel Nanda. Emergent misalignment is easy, narrow misalignment is hard. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=q5AawZ5UuQ.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning Gym: Reasoning environments for reinforcement learning with verifiable rewards. In *Advances in Neural Information Processing Systems 38 (NeurIPS), Datasets and Benchmarks Track*, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/528d56195a2c77c808494c86fa7c77ad-Abstract-Datasets_and_Benchmarks_Track.html.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 furious. arXiv preprint arXiv:2501.00656, 2025. URL https://arxiv.org/abs/2501.00656.

Ken Tsui. Self-correction bench: Uncovering and addressing the self-correction blind spot in large language models. In *Proceedings of the Conference on Language Modeling (COLM)*, 2026. URL https://openreview.net/forum?id=rLscoLa2TM.

Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13894–13908, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.826. URL https://aclanthology.org/2024.findings-acl.826/.

Jiaxin Wen, Ruiqi Zhong, Akbir Khan, Ethan Perez, Jacob Steinhardt, Minlie Huang, Samuel R. Bowman, He He, and Shi Feng. Language models learn to mislead humans via RLHF. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=xJljiPE6dg.

Nathan Whitehead and Alex Fit-Florea. Precision & performance: Floating point and IEEE 754 compliance for NVIDIA GPUs. NVIDIA whitepaper (2011), maintained as CUDA Toolkit documentation (v13.3), 2011. URL https://docs.nvidia.com/cuda/floating-point/index.html.

Xiaohu Xie, Xiaohu Liu, and Benjamin Yao. Know when you're wrong: Aligning confidence with correctness for LLM error detection. arXiv preprint arXiv:2603.06604, 2026. URL https://arxiv.org/abs/2603.06604.

Yuqing Yang and Robin Jia. When do LLMs admit their mistakes? Understanding the role of model belief in retraction. In *Proceedings of the Conference on Language Modeling (COLM)*, 2026. URL https://openreview.net/forum?id=QPHGIKoJKC.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*, 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. Understanding and mitigating numerical sources of nondeterminism in LLM inference. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*, pp. 188045–188077, 2025. doi: 10.52202/085713-5653. URL https://openreview.net/forum?id=Q3qAsZAEZw.

Haotian Zhai, Jingcheng Liang, and Dongyeop Kang. Abstain-R1: Calibrated abstention and post-refusal clarification via verifiable RL. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 19674–19695. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.985. URL https://aclanthology.org/2026.findings-acl.985/.

Jiachen Zhao, Yiyou Sun, Weiyan Shi, and Dawn Song. Can aha moments be fake? identifying true and decorative thinking steps in chain-of-thought. arXiv preprint arXiv:2510.24941, 2025. URL https://arxiv.org/abs/2510.24941.

Tianyi Zhao, Yinhan He, Wendy Zheng, Yujie Zhang, and Chen Chen. Wired for overconfidence: A mechanistic perspective on inflated verbalized confidence in LLMs. In *Proceedings of the Conference on Language Modeling (COLM)*, 2026. URL https://openreview.net/forum?id=68ah3EC3Iv.

A Experimental setup, models, and tasks

A.1 Models

Table 1 lists the model families used in the reported results. Our main families are Qwen2.5 (Qwen et al., 2024) and OLMo 2 (Team OLMo et al., 2025). Code and boundary experiments also use Llama 3 (Grattafiori et al., 2024), Phi-4 (Abdin et al., 2024), Gemma 2 (Gemma Team et al., 2024), and a DeepSeek-R1 distilled lineage (Guo et al., 2025). We use simple reader-facing names here.

A.2 Tasks

Table 2 summarizes the task domains. Countdown follows the arithmetic-search setup used in TinyZero and related small-scale reasoning studies (Pan et al., 2025; Gandhi et al., 2025). Shortest path comes from Reasoning Gym (Stojanovski et al., 2025), and the retained code boundary uses MBPP+ programming tasks (Austin et al., 2021).

The shortest-path task uses Reasoning Gym 0.1.25. Grids are 4–6 cells on each side, with blocked-cell probability 0.35; only solvable problems are kept, and a valid but non-shortest route counts

Table 1: Models used in the reported results. Boundary-only families are included only where they help define the scope of our claims.

Model	Size	Reporting setup	Role
Qwen2.5-1.5B	1.5B	Failure-report behavior installed through SFT	primary phenomenon, mechanism, and anchoring experiments
Qwen2.5-1.5B (second reference)	1.5B	Failure report + success pleasantry installed through SFT	independent second run set; success-side pleasantry control
Qwen2.5-1.5B (shortest-path reference)	1.5B	Failure-report behavior installed through SFT	second reasoning task; shortest-path anchoring
Qwen2.5-7B	7B	Failure-report behavior installed through SFT	larger-scale persistence experiment; 7B anchoring
Qwen2.5-14B	14B	Failure-report behavior installed through SFT	appendix execution-configuration boundary
Qwen2.5-32B-Instruct	32B	Instruction-conditioned; no report SFT	report-SFT-dependence boundary; not a matched scale-up
OLMo-2-1B	1B	Failure-report behavior installed through SFT	second model family; 20 retraining runs
Llama-3.1-8B	8B	Code self-report format; no failure-report SFT	code-domain boundary using verified test-execution reward and all-token KL

Table 2: Task domains and their role in the paper. Countdown and shortest path support the main claims; grid and code settings are targeted mechanism and boundary tests.

Task	What the model does	Automatic success check	Role in paper
Countdown	Use four numbers exactly once to build an arithmetic expression that reaches a target.	AST-parsed expression using +, –, ×, and ÷ only; exact rational evaluation; multiset number-use check	main setting for the primary runs, execution-configuration tests, mid-RL continuation tests, routing, and anchoring
Shortest path	Find a valid shortest path in a small generated graph or grid.	Reasoning Gym scorer, binarized at score = 1.0	second verifiable reasoning task; 20 retraining runs, anchoring, and shelter experiments
4×4 grid / constraint puzzles	Solve or assess small Latin-square / Shidoku-style constraint grids.	Exact solver oracle; every row, column, and when applicable box must be a permutation of 1–4	targeted pipeline tests of failure checking and report routing; not a co-equal replication
Code	Write Python functions for MBPP+ tasks and optionally report whether tests pass or fail.	True test-bank result from executed tests; self-report is ignored	boundary tests for direct reward, reward × reference anchoring, and surface/function separation

as a failure. The frozen dev set contains 100 problems from base seed 5101. The SFT reference is Qwen2.5-1.5B-Instruct fine-tuned on 1,800 traces, 15% of which contain the installed failure report. Its step-0 marginal disclosure on the nine-prompt probe is 9/75. Outcome-only RL uses 20 seeds, learning rate 10^{-6}, $G = 8$, 32 sequences per step, and 300 steps. Disclosure is measured on nine fixed prompts with 8×3 temperature-1 samples, while the headline solve result uses greedy evaluation on all 100 dev prompts. There is no shortest-path version of the report-entry intervention.

A.3 SFT WARM START AND POST-TRAINING

Most primary Countdown experiments begin from the same SFT reference model, trained on correct solutions together with examples that report and terminate on failure. We then apply KL-free outcome-only Dr. GRPO (Shao et al., 2024; Liu et al., 2025), which rewards verified task success but does not directly reward failure disclosure. This no-reference-KL choice is also used in modern reasoning-RL recipes such as DAPO (Yu et al., 2025); Section 5 later reintroduces reference regularization as the intervention. Table 3 summarizes the main training recipes, including the stabilized PPO boundary (Schulman et al., 2017).

The 1.5B reference is Qwen2.5-1.5B-Instruct fine-tuned on 1,000 Countdown records: 865 verified correct solutions and 135 constructed failure reports. Each failure example begins with a real model search on a modified task whose target cannot be reached. We stop the search after at least eight attempts, keep a verified near miss, and add a short ending that accurately reports the failure. The selected reference is the earliest checkpoint that passed the frozen format, solve, and reporting checks. The 7B reference uses the same corpus and selection rule, with the learning rate and memory layout adjusted for its size. The 32B experiment uses a deliberately different reporting setup. It starts directly from Qwen2.5-32B-Instruct, with no failure-report SFT; instead, a prompt instruction elicits failure reporting before outcome-only RL. We use this to test whether broad reporting variation depends on the report-SFT setup, not as a matched scale-up of the 1.5B and 7B experiments.

Each GRPO optimizer step contains 32 sampled sequences: four prompts, each with $G = 8$ completions, implemented as per-device batch 8×4 accumulation steps on one GPU in the default geometry. A 300-step run therefore samples 1,200 prompts and 9,600 completions from the 9,909-instance Countdown training split. The frozen Countdown dev set contains 100 separately generated problems. The training generator initially produced 10,000 problems; we removed 91 whose num-

Table 3: Main post-training recipes. The main outcome-RL runs use the same optimization scale; intervention rows change the reference term or algorithm as described in the text.

Setting	Post-training	Batch	Reward / reference term
Qwen-1.5B / Countdown (primary)	Dr. GRPO; 300 steps; $G = 8$; LR 10^{-6}	32 sequences = 4 prompts $\times$ 8 generations	Task reward; $\beta = 0$ (no reference KL)
Qwen-1.5B / Countdown (second run set)	Dr. GRPO; 300 steps; $G = 8$; LR 10^{-6}	32 sequences	Task reward; $\beta = 0$
OLMo-1B / Countdown	Dr. GRPO; 300 steps; $G = 8$; LR 10^{-6}	32 sequences	Task reward; $\beta = 0$
Qwen-1.5B / shortest-path	Dr. GRPO; 300 steps; $G = 8$; LR 10^{-6}	32 sequences	Task reward; $\beta = 0$
Qwen-7B / Countdown	Dr. GRPO; 300 steps; $G = 8$; LR 10^{-6}	32 sequences	Task reward; $\beta = 0$
Qwen-32B / Countdown (instruction-conditioned)	Dr. GRPO; 300 steps; $G = 8$; LR 10^{-6}	32 sequences (4 GPUs)	Task reward; $\beta = 0$
Qwen-1.5B / PPO boundary	Stabilized PPO; 300 batches; LR 10^{-6}	32 episodes / batch	Task reward; no reference KL

Table 4: Frozen evaluation surfaces for the main experiments. Stochastic probes use temperature 1.0 and a 1,024-token completion budget; greedy companions use temperature 0.

Setting	Frozen stochastic probe	Capability read
Primary Qwen2.5-1.5B / Countdown	20 pinned prompts $\times$ 8 samples $\times$ 3 probe seeds = 480 rows; vLLM (Kwon et al., 2023)	Correct / 480 on the same probe rows
OLMo-2-1B / Countdown	20 $\times$ 8 $\times$ 3 = 480 rows; HF decoder	Same probe rows; greedy dev-100 reported beside
Qwen2.5-1.5B / shortest-path	9 pinned prompts $\times$ 8 $\times$ 3 = 216 rows	Same probe rows; greedy solve on the 100 shortest-path dev prompts reported beside and used for headline selectivity
Qwen2.5-7B / Countdown	17 pinned prompts $\times$ 8 $\times$ 3 = 408 rows	Same probe rows; greedy dev-100 reported beside
Qwen2.5-32B-Instruct / Countdown	20 $\times$ 8 $\times$ 3 = 480 rows per prompt condition	Correct / 480 on the same probe rows

ber multiset and target matched any frozen evaluation problem, and every run checks this separation again at startup. Table 4 records the frozen evaluation surfaces used by the main experiments.

The Countdown probes are fixed, failure-heavy evaluation surfaces. For each reference model, we selected the frozen dev problems on which its greedy 1,024-token response was format-valid, incorrect, and matched the registered failure-report detector. We sorted these by task identifier and capped the set at 20. This produced 20 prompts for the 1.5B reference and 17 for the 7B reference; the sets are behavior-selected rather than random or difficulty-stratified samples of the dev set.

For the primary probe, disclosure is the number of detected reports divided by the number of terminated task failures. Terminated format-invalid completions remain in this denominator as non-disclosures, while unterminated completions are excluded. At step 300, format-invalid rows account for 11/9,600 probe rows across the 20 runs and unterminated rows for 24/9,600, so these cases are rare in the primary runs.

A.4 MID-RL STATE RESTORATION AND TERMINOLOGY

A retraining seed defines a standard training run, including the shuffled prompt schedule and rollout-sampling streams. An execution configuration changes low-level floating-point computation while

Table 5: What is restored in the exact-state mid-RL continuation experiments. These checks isolate future stochasticity without changing the earlier training history.

Component	What is held fixed or restored	Check
Model weights	Restored from the parent checkpoint.	Same-key continuations match endpoint and model artifacts.
Optimizer state	Adam moments are restored; the fork stops if state is missing.	Restoration is checked before continuation.
Scheduler / trainer step	Learning-rate scheduler and global training step are restored.	Continuation resumes from the parent step.
RNG state	CPU, CUDA, NumPy, and Python RNG states are restored; the continuation key then changes future rollout randomness.	The same key reproduces; different keys branch.
Data schedule	Parent prompt schedule and iterator position are held fixed.	All keyed continuations see the same training order.
Cross-host replay	The same saved state and continuation key are replayed on separate pods.	Bit-exact replay check gives endpoint and per-run-record agreement.

keeping the seed fixed. A continuation key is used only after restoring the full parent training state; it changes future rollout-sampling randomness while keeping the parent seed and data order fixed. Table 5 shows exactly what is restored and what source of randomness is changed.

A.5 Statistical estimands

The retraining run is the unit of analysis for variation across runs; prompts and sampled responses are never treated as independent retraining draws. We report raw standard deviation, SD ratios or log-SD ratios, hierarchical $\hat{\tau}$ where useful, and paired mean differences for matched interventions. For paired intervention arms, log-SD-ratio intervals use 20,000 paired resamples of the retraining draws, applying the same resampled indices to both arms. Intervals are 95% unless a pre-specified family uses the reported 97.5% or 98.33% interval. The hierarchical estimate $\hat{\tau}$ comes from a binomial logit-normal random-effects model with a 1,000-sample parametric bootstrap and is reported only for experiments with at least eight runs. If an interval includes the null, we describe the effect as "not detected", "not established", or "not distinguished", rather than treating it as evidence that the two conditions are equal.

B Measurement validation and qualitative examples

B.1 Failure-disclosure readouts

Our main Countdown measure, D_{v1}, is a conservative frozen detector for disclosure on terminated objective failures. It searches the reasoning span, without regard to letter case, for phrases such as "haven't found", "unable to find", "no exact", "closest", "nearest", "give up", "falls short", or "can't find/reach/land". It was fixed before the retraining runs were trained. Broader detectors add paraphrases, context checks, best-attempt statements, and answer-channel admissions. The first broader tier searches the final 250 characters for concession content; the next tier additionally searches the final 120 characters for statements about whether all numbers were used. We report these broader measures beside the primary detector rather than replacing it after seeing the outcomes. Table 6 summarizes the reader-facing readouts.

B.2 Qualitative output examples

The following three Countdown endings illustrate our failure-closing categories. The first explicitly acknowledges failure. The second ends on an objectively wrong answer without admitting failure or claiming success. The third falsely presents the attempt as successful with an incorrect verification.

Table 6: Reader-facing failure-disclosure measures. Internal detector version names and code paths are omitted here for readability.

Readout	What counts	Denominator	Use
Countdown disclosure (primary)	Explicit acknowledgment that the attempted Countdown solution failed.	Terminated task failures	main 1.5B Countdown measure; anchoring and mid-RL continuation experiments
Countdown disclosure (broad)	Broader semantic admissions, including paraphrases and answer-channel acknowledgments.	Terminated task failures	7B main readout and detector-sensitivity checks
Shortest-path disclosure	Domain-specific statements that no valid or shortest route was found.	Terminated shortest-path failures	shortest-path phenomenon and anchoring experiments
Route-entry disclosure	Broad semantic admission detector after removing the injected rider or prefill.	Failed continuations in the routing probe	P-B2 route-entry analysis
Auxiliary behavior detectors	Simple lexical or structured detectors for pleasantries and neutral codewords.	Setting-specific probe rows	P-D/P-E2/P-A/Kettle and formulation controls

> **Failure disclosure.** "... 18 - 5 = 13, 13 + 3 = 16, 16 - 19 = -3. That gives -3, which is not equal to 10. No exact solution turned up in this search. ..."
>
> **Silent wrong close.** "... 18 + 19 - 15 - 3 = 13. No. 19 - 18 + 5 + 3 = 14. No. ... `<answer>18 + 19 - 3 * 5</answer>`" The final expression evaluates to 22, but the ending neither admits failure nor claims success.
>
> **Fabricated success.** "... 18 - 5 = 13, 13 - 3 = 10. Yes! Verify: 18 - 5 = 13, 13 - 3 = 10. ..." The emitted answer is `18 - 5 - 3 + 5`, which evaluates to 15 and also uses the numbers incorrectly.

An observable multi-stage example. The grid setting provides an example in which several stages are visible in the generated text. The model first proposes a candidate grid, then checks its entries, identifies a repeated value, forms the failure verdict that the grid is invalid, and finally states: *"The check shows my grid is wrong."*. This separates candidate generation, explicit checking, a grounded failure verdict, and semantic admission. It does not reveal whether the model had any unexpressed recognition before writing the check.

B.3 Blind audits and detector sensitivity

Table 7 reports the human and independent model-reader checks separately. The human audit covered a broad sample rather than only the most extreme cases: after removing duplicates, it included 552 examples across seven sampling strata. A fixed validation subset contained 98 examples, 97 of which received a human label. The detector agreed with the human label on 80/97 examples (82.5%), including 40/42 examples in the concession strata, with no cases where the detector marked a concession that the human reviewer judged absent. The reported $\kappa = 0.822$ is instead agreement between two independent model readers and should not be interpreted as human agreement. A separate re-read of 48 endings sampled low-, middle-, and high-disclosure runs, then randomly sampled examples within those runs. Table 9 shows results under broader detector definitions, and Table 10 reports the calibrated instruction-conditioned 32B results.

We also use format-specific labels to distinguish explicit admissions of failure, incorrect answers that end without an admission, and incorrect answers that are presented as successful. As a sanity check, we apply this taxonomy to verifier-correct outputs from the primary 1.5B runs. It rarely labels a correct answer as an admission: the false-alarm rate is 0–5.2% across runs and 1.5% overall

Table 7: Summary of the human and independent model-reader audits. The two audit types are reported separately.

Audit	Setting	Sample	Main result
Human semantic-label audit	Seven analysis strata	560 sampled; 552 unique rows; 97/98 validation rows scored	Detector–human agreement 80/97 (0.825); 40/42 in concession strata, with no human false positives. Model–model $\kappa = 0.822$.
Human blind re-read	Primary / second run set / installed-channel runs	48 real endings from low, high, and median runs; random rows within runs	44/48 agreement; Cohen kappa 0.846 [0.678, 0.965].
Human second rater	Independent ending labels	40 endings, seeded stratified sample	39/40 agreement; Cohen kappa 0.950.
Model-reader audit	1.5B anchoring arms	50 positive + 50 negative per arm	Cohen kappa 0.941–1.000; 0/50 false positives in all three arms.
Model-reader audit	Shortest-path runs	50 + 50 per horizon	Cohen kappa 1.000 @150 and 0.961 @300; no misses or false positives.
Model-reader audit	7B common-path runs	50 + 50 per horizon	Cohen kappa about 0.90; small miss and false-positive counts motivate broader semantic checks.

Table 8: How failed responses end with and without anchoring at step 300. False success means that the model presents a wrong answer as correct. Entries are the mean across runs, followed by the range across runs in parentheses, and are calculated as shares of terminated failures. Responses that do not reach a normal termination before the generation limit are reported separately and excluded from these shares. The 1.5B and 7B rows use separate fixed classifiers suited to their response formats, so comparisons involving anchoring are made within each model setting.

Setting	n	False success	Silent close	Admission	False-success share of nondisclosure
1.5B, no anchor	20	0.18 (0.09–0.28)	0.14 (0.00–0.51)	0.69 (0.37–0.91)	0.67
1.5B, failure-conditional anchor	20	0.18 (0.08–0.28)	0.10 (0.00–0.32)	0.72 (0.49–0.83)	0.66
1.5B, all-token anchor	20	0.16 (0.09–0.21)	0.12 (0.04–0.29)	0.72 (0.58–0.82)	0.60
7B, no anchor	19	0.27 (0.02–0.59)	0.21 (0.00–0.46)	0.52 (0.04–0.94)	0.60
7B, failure-conditional anchor	19	0.14 (0.03–0.28)	0.02 (0.00–0.06)	0.84 (0.71–0.97)	0.89

(62/4,010 correct completions). These labels are used only for the format-specific analyses and are separate from the paper's primary lexical disclosure detector.

Looking at Table 8, in the primary 1.5B comparison, failure-conditional anchoring does not detectably change the average rate of false success claims or silent closes, although the spread of silent closes narrows. At 7B, anchoring lowers false success claims by 0.127 [95% CI: 0.059, 0.194] and silent closes by 0.191 [0.101, 0.281] across the 19 matched seeds; the spread of both behaviors also narrows. Anchoring therefore reduces both forms of nondisclosure at 7B, but it removes silence more completely than false success claims. Such non-terminating responses are rare: they average at most 0.7% of trajectories in the 1.5B runs and 1.4% in the unanchored 7B runs.

Table 9: Sensitivity to broader detector definitions. Broader semantic detectors change the estimated levels somewhat, but the large differences across retrainings remain. Broadest SD uses the union of all seven detector tiers for the 1.5B rows and the broadest single semantic detector for the 7B row.

Run set	Primary SD	Broader SD	Broadest SD	Main result
Primary Qwen-1.5B	0.185	0.163	0.113	Minimum rank agreement across nested detector tiers is $\rho = 0.908$.
Second Qwen-1.5B	0.220	0.175	0.169	Minimum rank agreement across nested detector tiers is $\rho = 0.917$.
Qwen-7B	0.281	0.255	0.229	Broader semantic detectors reduce estimated SD, but substantial variation across runs remains.

Table 10: Calibration for the instruction-conditioned Qwen2.5-32B boundary experiment, which elicits failure reporting through the prompt without report SFT. The frozen-detector lower bound and the blind-read calibrated estimate are both stated in the main text.

Condition	Unit	Calibrated disclosure [95% bootstrap interval]	Solve
Instruction prompt, before RL	1 parent	0.641 [0.593, 0.688]	0.288
No instruction, before RL	1 parent	0.000	0.215
Instruction-conditioned retrainings @300	8 draws	0.173–1.000	0.302–0.619

Table 11: Across-retraining ranges and raw SDs for the main phenomenon and generalization experiments. These rows are descriptive and are not pooled into a single universal effect.

Setting	n	Disclosure range	Solve range	SD(disclosure)	SD(solve)
Qwen-1.5B / Countdown	20	0.257–0.903	0.340–0.490	0.185	0.047
Qwen-1.5B / second run set†	20	0.171–0.895	0.530–0.740	0.220	0.063
OLMo-1B / Countdown	20	0.072–0.903	0.375–0.492	0.255	0.034
Qwen-1.5B / shortest-path*	20	0.000–0.986	0.460–0.610	0.349	0.044
Qwen-7B / Countdown	19	0.000–0.770	0.150–0.561	0.281	0.122
Qwen-32B / instruction-conditioned	8	0.056–0.628	0.302–0.619	0.238	0.106
Qwen-1.5B / PPO	8	0.673–0.814	0.227–0.263	0.047	0.013

*Shortest-path solve uses greedy evaluation on the 100 shortest-path dev prompts; disclosure uses the 216-row stochastic probe. †Second-run-set solve uses greedy evaluation on the 100 Countdown dev prompts. The 32B row is an instruction-conditioned boundary test without report SFT; its disclosure range uses the frozen-detector lower bound, while blind-read calibration gives 0.173–1.000.

C Primary distributions, generality, and recipe boundaries

Tables 11 and 12 summarize the main statistics across runs. For the primary runs, solve and disclosure are not detectably correlated across retrainings (Pearson $r \approx -0.03$). For the second Qwen experiment, the hierarchical disclosure estimate is $\hat{\tau} = 0.949$ [0.613, 1.266].

C.1 Primary retrainings: robustness to evaluation noise and probe selection

The large disclosure spread is not explained by finite sampling on the registered probe. After accounting for finite-probe binomial noise on the logit scale, the primary runs retain substantial disclosure heterogeneity ($\hat{\tau} = 0.864$ [0.563, 1.129]), while solve is about 0.166. We report the solve value as a level rather than an interval because its fitted $\hat{\tau}$ is more sensitive to repeated-prompt clustering corrections. Across the frozen evaluation ladder, between-run disclosure SD grows from 0.061 at

Table 12: Headline selective-dispersion estimates and 95% uncertainty intervals. SD ratios compare across-run disclosure SD with solve SD; the second Qwen experiment instead uses the hierarchical $\hat{\tau}$ ratio.

Setting	n	Disclosure $\hat{\tau}$ [95%]	Selective-dispersion statistic	Estimate [95%]
Qwen-1.5B / Countdown	20	0.864 [0.563, 1.129]	SD(disclosure) / SD(solve)	3.93 [2.64, 5.50]
Qwen-1.5B / second run set[†]	20	0.949 [0.613, 1.266]	$\hat{\tau}$(disclosure) / $\hat{\tau}$(solve)	5.67 [2.89, ∞]
OLMo-1B / Countdown	20	1.22 [0.81, 1.58]	SD(disclosure) / SD(solve)	7.51 [5.40, 10.71]
Qwen-1.5B / shortest-path[*]	20	2.81 [1.88, 3.80]	SD(disclosure) / SD(greedy solve)	7.89 [5.20, 11.55]
Qwen-7B / Countdown	19	2.28 [1.47, 3.06]	SD(disclosure) / SD(solve)	2.30 [1.55, 3.94]
Qwen-1.5B / PPO	8	–	SD(disclosure) / SD(solve)	3.55 [1.35, 7.63]

[*]Greedy solve on the 100 shortest-path dev prompts; the floor-adjacent stochastic-probe companion is discussed in text. [†]The second run set's $\hat{\tau}$(solve) uses greedy evaluation on the 100 Countdown dev prompts.

Table 13: Disclosure trajectory for the primary runs on the fixed stochastic probe. The reference-model row is evaluated on the same probe; the remaining rows summarize the 20 retrainings.

Checkpoint	Mean disclosure	Across-run SD
SFT reference	0.677	–
Step 15	0.691	0.061
Step 30	0.697	0.055
Step 45	0.707	0.057
Step 60	0.708	0.095
Step 90	0.685	0.100
Step 120	0.657	0.104
Step 150	0.666	0.139
Step 210	0.678	0.132
Step 300	0.628	0.185

Table 14: Capability dispersion on the registered probe and on the full 100-prompt dev set. The full-dev read for the primary runs is a post-hoc, inference-only robustness check; the registered 480-row probe remains primary. The simple binomial floor is a finite-sample reference, not a formal null test.

Capability evaluation	Prompts per run	Solve range	Cross-run SD	Binomial floor
Primary registered stochastic probe	20 $\times$ 8 samples $\times$ 3 seeds	0.34–0.49	0.047	–
Primary post-hoc greedy dev set, H200	100	0.65–0.78	0.045	0.045
Second Qwen run set, greedy dev set	100	0.53–0.74	0.063	0.047

step 15 to 0.185 at step 300, while the mean disclosure rate remains between 0.628 and 0.708. The step-150 ordering predicts step 300 poorly (Spearman $\rho = 0.14$ [-0.34, 0.56]), so the spread is not well described as uniform decay from the SFT reference. Table 13 gives the full trajectory.

Table 15: Post-hoc greedy disclosure on the full dev set and its selected versus unselected prompt subsets. The selected and unselected rows use smaller, run-dependent failure denominators and are shown only to test whether behavior-based probe selection inflated the spread.

Surface	Prompts	Disclosure range	Across-run SD	Hierarchical $\hat{\tau}$
Full dev set, primary detector	100	0.13–0.93	0.235	1.03 [0.63, 1.40]
Full dev set, broadest detector	100	0.39–0.93	0.169	0.69 [0.34, 0.96]
Selected probe prompts, primary detector	20	0.09–1.00	0.251	–
Other dev prompts, primary detector	80	0.12–0.88	0.266	–

C.2 Smaller recipe and family boundaries

Table 16: Selected training-recipe, algorithm, and model-family boundaries. These are supporting scope checks rather than headline replications.

Boundary	Setting	n	Main result
Group size	G=16, @150	n=3	Disclosure spread is 0.419; variability remains, but this arm also changes prompt exposure per step.
Larger batches	Batch 64/128	n=3 each	Broad disclosure variation remains, but endpoint capability is confounded; a capability-matched companion remains above the prespecified spread floor.
Gemma-2-2B	Countdown	n=3	Does not reproduce the broad primary-detector pattern at this small n; the reporting surface also differs.
Llama-3.2-3B	Countdown	n=3	Disclosure spread is 0.194; we make no broad-variability claim at this small n.
Distilled Qwen-1.5B	Countdown	n=3	Tight boundary: disclosure spread is 0.076 at step 300 and 0.129 at step 150.
Phi-4	Countdown	n=3	Disclosure stays uniformly high while capability varies substantially; capability-confounded boundary.
RLOO	Countdown	n=2	Descriptive comparison that is sensitive to execution configuration and hardware class; not evidence of algorithm invariance.

A broader capability evaluation gives the same conclusion. On the complete 100-prompt dev set using the H200 venue of record, greedy solve ranges from 0.65 to 0.78 across runs with SD 0.045, equal to the estimated finite-sample floor. A repeat on a 3090 Ti yields the same dispersion conclusion (solve SD 0.054), although only 48% of greedy responses are byte-identical across GPU classes. Thus the relatively stable capability result is not an artifact of repeatedly sampling the 20 registered probe prompts. Table 14 reports the full comparison.

The disclosure spread also persists outside the behavior-selected probe. On the same 100-prompt H200 evaluation, disclosure among each run's naturally occurring failures spans 0.13–0.93 under the primary detector (SD 0.235; $\hat{\tau} = 1.03$ [0.63, 1.40]). Restricting to the 80 prompts that were not used to construct the registered probe still gives a range of 0.12–0.88 (SD 0.266), compared with SD 0.251 on the selected 20 prompts. Thus the broad reporting variation is not created by selecting prompts on which the SFT reference disclosed. Because different retrainings fail on different subsets of the dev set, these are descriptive rates on each run's own failure set rather than paired prompt-level effects. Table 15 gives the full results and detector sensitivity.

Table 17: Six micro-batch × gradient-accumulation execution configurations with the seed and global batch size fixed. The disclosure column uses the pre-specified endpoint measure shown in Figure 3A.

Micro-batch / accumulation	Micro-batch	Grad. accum.	Disclosure	Solve
pd1ga32	1	32	0.186	0.463
pd2ga16	2	16	0.519	0.354
pd4ga8	4	8	0.836	0.435
pd8ga4	8	4	0.787	0.412
pd16ga2	16	2	0.848	0.365
pd32ga1	32	1	0.644	0.327

For shortest-path, the headline capability comparison uses greedy solve on the 100 shortest-path dev prompts (0.46–0.61 across runs), giving a disclosure/solve SD ratio of 7.89 [5.20, 11.55]. The separate 216-row failure-heavy stochastic probe has solve near floor, which mechanically limits its raw solve dispersion; its raw SD ratio is 6.29 [4.72, 8.69], while the noise-adjusted logit-scale $\hat{\tau}$ ratio is not established (1.47 [0.87, 2.72]).

A two-seed RLOO comparison is included only as a descriptive boundary because the sample is too small and the result is sensitive to the execution configuration and hardware class. The stabilized-PPO experiment is the main algorithm boundary discussed in the paper. It shows the same direction with a smaller absolute spread (disclosure 0.673–0.814 across the eight runs; SD 0.047 versus 0.013 for solve; Table 11); because the PPO runs also sit in a lower solve regime (0.227–0.263), we cannot attribute the difference in absolute magnitude to the RL algorithm alone.

D Execution configurations and mid-RL continuations

D.1 Execution configurations and replay checks

Figure 3A and Table 17 show the six pre-specified 1.5B execution configurations and their step-300 probe measurements. We also record the on-policy training-rollout give-up share in 15-step windows, but that diagnostic uses a different prompt distribution and a different denominator: it counts give-ups among all training completions, whereas the primary endpoint counts disclosures among failed probe responses. The training curves can therefore remain relatively close even when the fixed-probe disclosure endpoints separate widely, so we do not combine the two measures in one panel. Table 18 separates exact same-configuration replay from changes to the hardware or GPU configuration. Across the six splits, every micro-batch is equal-sized, gradient accumulation applies the corresponding accumulation-step scaling, and dropout is inactive by model configuration. Under the Dr. GRPO loss used here, each completion therefore has the same $1/(32 \cdot 1024)$ weight in the accumulated gradient regardless of the split, with group-mean advantages computed before splitting. The six configurations optimize the same objective with the same exact-arithmetic gradient and differ only in floating-point kernel shapes and reduction order. A one-step identical-state check gives update cosine 0.99982 and relative difference 0.019, consistent with rounding-level floating-point differences rather than sequence reweighting.

A larger-model example supports the same execution-configuration interpretation, although it is not a clean replication. For Qwen2.5-14B under a separate length-penalty recipe, four low-level execution/hardware configurations produce step-300 disclosure values of 0.125, 0.533, 0.874, and 0.933. Greedy solve is 0.82–0.90 on the three configurations with committed solve measurements. We keep this result in the appendix because the configuration manipulation combines hardware and world-size differences and the recipe adds a length penalty. We thus use it only as supporting evidence for sensitivity to low-level execution details, not part of the main six-configuration estimate.

D.2 Continuations from shared mid-RL states

Figures 5 and 6 show the full parent-by-continuation results. Table 19 summarizes the endpoint ranges. The branch-and-continue replication contains eight ten-continuation cells: four parent mid-

Table 18: Replay and hardware/GPU checks. Repeating the same execution configuration is bit-exact in our B200 setup; changing hardware class can change generation from the first training step.

Check	Comparison	Result
Same execution configuration	5 replays across two B200 hosts / GPU indices	Group logs, probe samples, and fixed evaluation windows are bit-exact in 5/5 replays.
Micro-batch re-split	Same seed; different micro-batch / accumulation split	Step-1 rollout generation is identical; later numerical updates cause the training records to diverge.
Hardware/GPU path	B200 vs H200 path	Generation differs from step 1; endpoint disclosure is 0.787 vs 0.204 at similar solve.

RL states evaluated at branch points 150 and 240, with four admission cells and four codeword-emission cells. Their within-cell rate SDs range from 0.07 to 0.27, with a median of 0.10. The step-150 disclosure range of 0.11–0.88 shown in the main text is the widest cell in this replication rather than the only mid-RL state that shows substantial divergence.

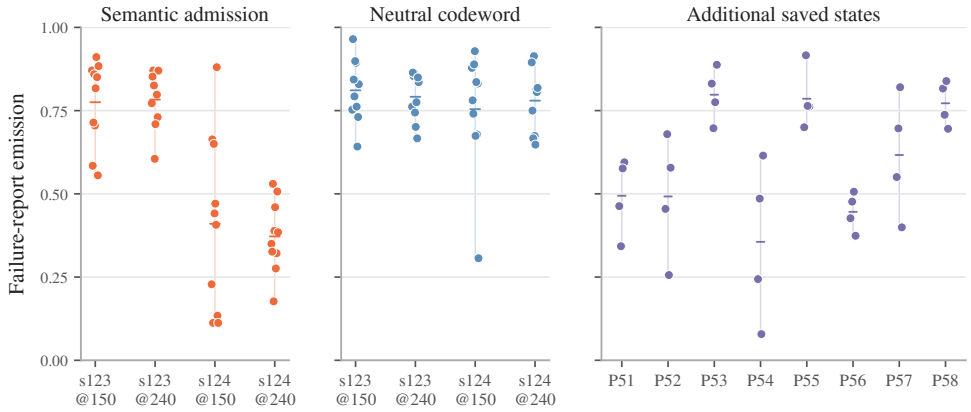

Figure 5: Failure-report emission across continuations from eight shared mid-RL states. Each column starts from one identical saved parent state; the dots differ only in future rollout randomness. Short horizontal ticks show means, and faint vertical lines show the observed min–max range.

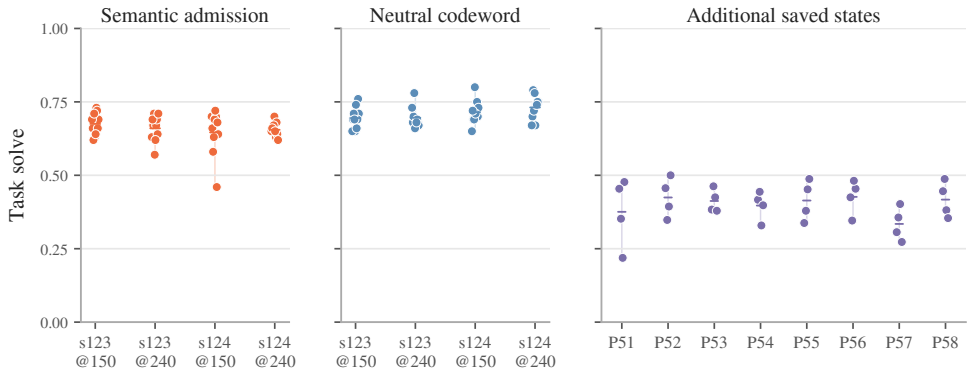

Figure 6: Task solve for the same mid-RL continuations. In the cells with the widest reporting variation, task capability varies much less.

Table 19: Endpoint ranges across continuations from the shared mid-RL states. Admission and post-history rows report failure disclosure; codeword rows report codeword emission. Solve is greedy dev-100 solve for the admission and codeword rows and probe solve (correct/480) for post-history.

Saved state	Keys	Measured rate @300	Solve @300
admission 1, s123 @150	10	0.556–0.911	0.62–0.73
admission 1, s123 @240	10	0.605–0.871	0.57–0.71
admission 1, s124 @150	10	0.112–0.880	0.46–0.72
admission 1, s124 @240	10	0.177–0.530	0.62–0.70
codeword 1, s123 @150	10	0.642–0.965	0.65–0.76
codeword 1, s123 @240	10	0.667–0.864	0.66–0.78
codeword 1, s124 @150	10	0.307–0.929	0.65–0.80
codeword 1, s124 @240	10	0.648–0.914	0.67–0.79
post-history parent 51	4	0.343–0.595	0.22–0.48
post-history parent 52	4	0.256–0.679	0.35–0.50
post-history parent 53	4	0.697–0.888	0.38–0.46
post-history parent 54	4	0.079–0.615	0.33–0.44
post-history parent 55	4	0.700–0.916	0.34–0.49
post-history parent 56	4	0.374–0.506	0.35–0.48
post-history parent 57	4	0.399–0.820	0.27–0.40
post-history parent 58	4	0.695–0.839	0.35–0.49

E FAILURE-REPORTING PIPELINE AND ROUTING

E.1 PIPELINE EVIDENCE MAP

Table 20 maps the retained experiments onto the stages of failure reporting and also states what each experiment does *not* establish.

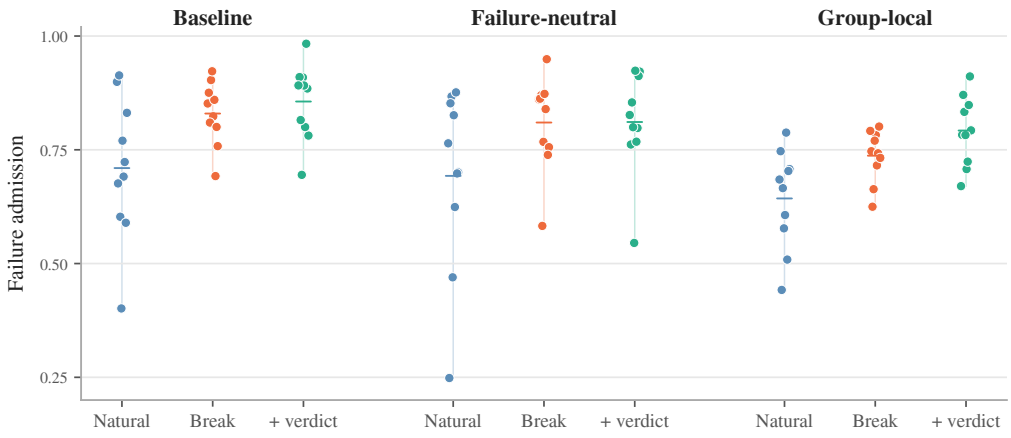

Figure 7: Natural continuation, the paragraph-break cue after the last attempt (Break), and the same seam with an explicit true verifier verdict at step 300. Neither intervention supplies report words. Break raises later admission and reduces between-run spread in the primary recipe; the NoBreak control in Table 22 shows that most of this tightening comes from the inserted paragraph cue rather than re-entry alone. Providing a verifier verdict of failure produces no additional tightening, although it raises mean admission in the group-local variant. Short horizontal ticks show means, and faint vertical lines show observed min–max ranges.

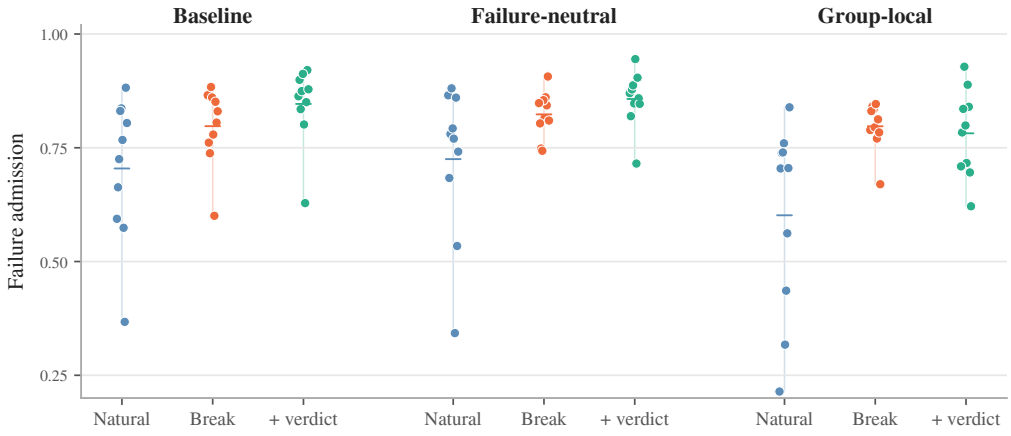

Figure 8: The same natural/paragraph-break/verdict-at-the-seam experiment at the step-150 companion horizon.

Table 20: How retained experiments map onto the stages of failure reporting. The final column states each result's main limitation, avoiding an overly simple one-experiment-per-stage interpretation.

Stage	Evidence	What it supports	Main limitation
Failure evidence / checking	Grid/constraint checks	The visible report can separate from the computation that supports a failure verdict.	Grid-only evidence; not a Countdown localization result.
Report-entry template cue	NoBreak versus Break at the same seam, 1.5B	Most tightening comes from supplying the paragraph break that begins the learned failure-report template, not from re-entry alone.	Format-specific cue; does not reveal whether the model internally detected the failure.
Natural transition probability	Teacher-forced break versus report-entry comparison	The break and the next report-entry step are not distinguished in the three step-300 comparisons; the inserted break is strongly off-policy.	Correlational over ten runs per cell; does not identify cause or internal recognition.
Inline initiation at 7B	Two low-disclosure 7B runs	Both reach the reporting point; supplying enough of the inline admission opening yields 0.97 / 1.00 acknowledgment.	Two runs; not a population-level rate.
Failure cue versus forced initiation	Verdict-only B/ORACLE; two-word and first-third prefixes C/D	Verdict-only gives no consistent benefit; forced prefixes raise admission much more.	An inserted verdict need not reproduce the model's own recognition state.
Route decomposition	Route-decomposition probe (P-B2)	Route entry explains 70% [49, 91] of raw variation; downstream persistence explains 23% [14, 32].	Downstream variation remains; the relative-logit comparison is not distinguished.
Surface vs function	Codeword control	Keeping a reporting-related string does not guarantee that its connection to failure is preserved.	Boundary evidence, not a complete stage-by-stage causal map.

E.2 ROUTING LOCALIZATION

Table 21 summarizes the failure-to-report (F2R) seam re-entry and report-prefix analysis, which intervenes at the point where a failure report could begin, together with the three-arm natural/paragraph-break/verdict-at-the-seam probe and the route-decomposition probe (P-B2). Here the *reporting point* is the end of the attempted solution where a failure report could begin. Figures 7 and 8 show the full natural/paragraph-break/verdict distributions at both training horizons; Table 22 gives the direct Natural–NoBreak–Break comparison.

Table 21: Summary of the seam, paragraph-cue, report-prefix, and route-decomposition analyses. The no-inserted-characters control shows that most of the earlier localization effect comes from the learned paragraph-break cue rather than re-entry alone.

Experiment	Scope	Main result
Natural reach, 1.5B	40 checkpoint/formulation combinations	Median reach is 0.952; reach vs natural disclosure $\rho = 0.098$. The conditional-admission correlation is largely expected because reach is near one.
Nothing-inserted re-entry (NoBreak)	Same seam; no inserted characters	Admission rises slightly, but between-run spread changes little: the hierarchical contrast versus Natural is +0.02 to +0.21 and is detected in 2/6 cells.
Paragraph-break cue (F2R A / CTRL)	Same seam; two newlines; no report words	Tighter than NoBreak in all six point estimates; the contrast is detected in 3/6 cells on the hierarchical scale and all six on the raw scale. Most of the earlier localization effect comes from the learned paragraph cue.
Teacher-forced transition check	Same 60 checkpoints and failed trajectories	At step 300, variation in the break probability is not distinguished from variation in report entry after the break. The natural probability of the break is only about 0.00015–0.0008.
Verdict at the seam (F2R B / ORACLE)	Same seam; no report words	No consistent level gain and no additional tightening. One recipe variant has a +0.055 mean change.
Forced report prefix (F2R C / D)	Two words / first third of phrase	Admission rises by 0.141 and 0.167 over paragraph re-entry alone; 39/40 and 40/40 combinations increase.
Route-decomposition probe (P-B2)	n=20	Raw split: entry 70% [49, 91], persistence 23% [14, 32]. The ordering does not hold on the logit scale because persistence is near ceiling.
Inline-format 7B check	Two low-disclosure runs	Both reach the reporting point; supplying enough of the inline admission opening yields genuine acknowledgment in 0.97 / 1.00 of continuations.

The paragraph-break condition (F2R A), which also serves as the control (CTRL) in the three-arm probe, supplies no report words. It cuts the trajectory immediately after the model's last attempt sentence, inserts the two-newline paragraph break used by the SFT failure-report template, and

Table 22: No-inserted-characters control. Positive values in the first contrast mean that Break is tighter than NoBreak; positive values in the second mean that NoBreak is tighter than Natural. Intervals are 95%.

Cell	$\ln(\hat{\tau}_{\mathrm{NoBreak}}/\hat{\tau}_{\mathrm{Break}})$	$\ln(\hat{\tau}_{\mathrm{Natural}}/\hat{\tau}_{\mathrm{NoBreak}})$	Mean admission: Natural / NoBreak / Break
Primary @300	+0.45 [+0.07, +1.02]	+0.08 [−0.02, +0.20]	0.710 / 0.761 / 0.830
Primary @150	+0.38 [+0.11, +0.74]	+0.08 [+0.002, +0.18]	0.704 / 0.746 / 0.797
Failure-neutral @300	+0.26 [−0.36, +0.80]	+0.08 [−0.07, +0.14]	0.693 / 0.736 / 0.810
Failure-neutral @150	+0.64 [−0.08, +1.17]	+0.21 [+0.04, +0.30]	0.725 / 0.762 / 0.823
Group-local @300	+0.42 [−0.05, +1.20]	+0.10 [−0.15, +0.21]	0.643 / 0.676 / 0.737
Group-local @150	+1.16 [+0.90, +1.95]	+0.02 [−0.12, +0.09]	0.602 / 0.659 / 0.797

draws four fresh continuations. Relative to the natural endpoint, this condition raises admission in 36/40 checkpoint–formulation combinations. At step 300, raw across-run admission SD falls from 0.156 to 0.069 in the primary recipe and from 0.201 to 0.102 in the second recipe. The raw SD contrast favors tightening in all six recipe-by-horizon cells. On logit and hierarchical scales that account for the higher mean, tightening is established in the primary recipe at both horizons and in the sibling recipe at step 150.

The registered NoBreak control separates re-entering the seam from inserting the paragraph cue. It uses the same 60 checkpoints, failed trajectories, seam, and four-continuation instrument, but appends no characters. Re-entry alone raises mean admission by 0.03–0.06 across the six cells and reduces hierarchical spread only slightly: $\ln(\hat{\tau}_{\mathrm{Natural}}/\hat{\tau}_{\mathrm{NoBreak}})$ ranges from +0.02 to +0.21 and is detected in two cells. The paragraph break adds another 0.05–0.14 in mean admission and tightens substantially beyond NoBreak: $\ln(\hat{\tau}_{\mathrm{NoBreak}}/\hat{\tau}_{\mathrm{Break}})$ ranges from +0.26 to +1.16, is detected in three of six hierarchical contrasts, and in all six raw-SD contrasts. Thus most of the earlier stabilization comes from supplying the learned paragraph transition rather than re-entering the seam itself.

The intervention has two remaining caveats. First, the paragraph transition is specific to the installed response template: the located seam matches where the report phrase naturally begins in 572/2,502 (23%) of naturally disclosing rollouts. Second, the seam conditions use four fresh samples per trajectory, whereas the natural endpoint uses one continuation. The estimated binomial noise floor is therefore about 0.013 under seam continuation and 0.027 under natural continuation. Subtracting those floors still leaves SDs of about 0.068 and 0.154 for Break and Natural in the primary comparison, so the sampling difference does not explain the observed halving.

A teacher-forced check on the same 60 checkpoints asks whether runs differ more in producing the paragraph break or in entering the report after the break is supplied. The analysis compares these probabilities on a logit scale and keeps each recipe and horizon separate. At step 300, the two stages are not distinguished in any of the three recipes. In the primary recipe at step 150, the break probability is more variable, but it has almost no relationship with natural disclosure ($\rho = -0.01$), and the result does not repeat in the two related recipes. Table 23 reports all six comparisons.

The model's own probability of producing the paragraph break at this seam is only about 0.00015–0.0008. Supplying the break is therefore a strong off-policy cue, not a typical natural continuation. A descriptive pattern appears one token later: across all six cells, runs with more natural disclosure place more probability on an admission-first opening and less on a re-check opening. The total probability of entering the report paragraph does not increase consistently. These associations use ten runs per cell and have no uncertainty intervals, so we treat them as a description of where differences are visible under teacher forcing, not as evidence of cause or internal failure recognition.

Table 23: Teacher-forced comparison of the learned paragraph transition with the next report-entry step. Positive log SD ratios mean that the break probability varies more across retrainings than the total probability of a report opening after the break is supplied. The primary step-300 comparisons do not distinguish the stages.

Setting	Step	Log SD ratio [95% interval]	ρ(break, natural disclosure)	Reading
Primary	300	+0.421 [−0.356, +1.254]	−0.25	Not distinguished
Failure-neutral	300	+0.614 [−0.036, +1.390]	−0.24	Not distinguished
Group-local	300	+0.526 [−0.196, +1.175]	+0.41	Not distinguished
Primary	150	+0.930 [+0.332, +1.441]	−0.01	Break probability is more variable, but does not track natural disclosure
Failure-neutral	150	+0.214 [−0.468, +0.663]	−0.89	Not distinguished
Group-local	150	+0.464 [−0.135, +1.012]	−0.37	Not distinguished

Verdict-only evidence already exists. F2R condition B inserts a failure cue at the same seam without report words; relative to condition A it changes admission by +0.027, with 31 increases, 9 decreases, and sign-flip $p = 0.159$, while the functional read changes by -0.006. In the three-arm probe, ORACLE adds the true verifier verdict to the no-words paragraph re-entry. At step 300 its mean changes relative to CTRL are +0.026 [-0.014, +0.067] in the primary recipe, +0.001 in the second recipe, and +0.055 [+0.023, +0.087] in the group-local variant; none of the six recipe-by-horizon comparisons tightens further. In contrast, F2R conditions C and D supply a two-word prefix or the first third of the report phrase and raise admission over condition A by 0.141 (39/40 combinations) and 0.167 (40/40). Thus failure evidence alone is insufficient in this interface, while forced report initiation has a larger effect.

The natural-output route decomposition provides a descriptive companion without seam intervention. Route entry is the fraction of terminated failed rows that match the frozen semantic-concession detector after the injected rider is removed. Persistence is the fraction of those semantic entries that retain the rider. Route entry ranges from 0.22 to 0.91 across runs; persistence ranges from 0.69 to 1.00, with median 0.97. A raw-scale Shapley decomposition of the unconditional rider rate assigns 70% [49, 91] of its variation to route entry, 23% [14, 32] to persistence, and 7% to off-route use. Because persistence is near ceiling, this ordering does not survive a mean-adjusted analysis: at step 300 the hierarchical logit-scale spread is 0.960 [0.636, 1.240] for route entry and 1.356 [0.818, 1.861] for persistence, and the two are not distinguished. At step 150, persistence is more dispersed on the logit scale. We therefore do not use the raw 70% split as separate mechanistic proof. The experiments also do not show whether naturally silent failures went undetected or were detected but not reported, because an inserted verdict need not reproduce the model's own recognition state.

All four 1.5B Countdown formulations share one SFT format in which the failure report begins as a new paragraph after the attempted solution. The NoBreak result shows that the main 1.5B localization effect is substantially a learned paragraph-template cue, not a universal report-initiation token. The 7B lineage instead reports inline: two low-disclosure runs still reach the reporting point and acknowledge failure in 0.97 and 1.00 of continuations once enough of the admission opening is supplied. A grid interface with single-line disclosures provides another format boundary. The initiating move can thus be a paragraph, phrase, or token sequence depending on lineage and interface. These checks are supporting evidence rather than co-equal replications of the main localization result.

F Neutral, pleasantry, and formulation controls

Our controls use three different kinds of text. The weakly constrained (P-D) and explicit-cue (P-E2) experiments both use the fixed phrase "Thanks for the question." The success-side pleasantry (P-A) is a family of polite closing sentences associated with solved SFT examples, rather than one exact string; examples include variants such as "That was a fun one to think about—thank you very much for the nice puzzle." The neutral-codeword/phrase result in Figure 3E uses Kettle: "A copper kettle rests on the kitchen shelf."

Table 24: Main neutral and pleasantry controls. The first four provide context for Figure 3E; P-C is kept only as an appendix structural boundary.

Control	n	Step-300 range	Interpretation
Weakly constrained neutral phrase (P-D)	20	0.00–0.90	Large variation without failure-related meaning.
Explicit deterministic cue (P-E2)	20	0.993–1.000	19/20 runs are exactly 1.0 at step 300; same phrase and corpus as P-D.
Success-side pleasantry (P-A)	20	0.796–1.000	Stable success-side control; 13/20 runs are exactly 1.0 at step 300.
Kettle failure-conditioned neutral codeword	20	0.59–0.94	Pooled 20-run codeword control used in Figure 3E.
Failure-branch inert filler (P-C)	20	0.31–0.90	Appendix-only structural boundary; does not causally isolate branch location.

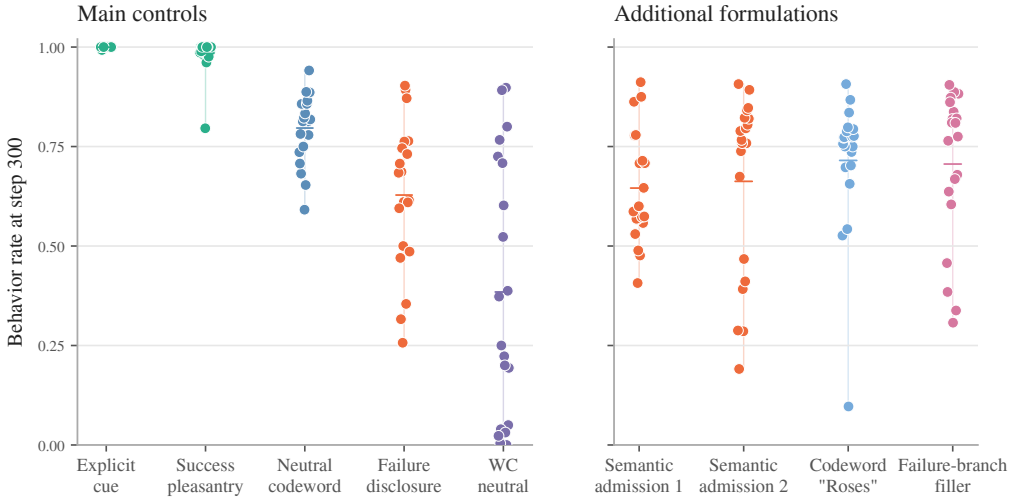

Figure 9: Broader determination/susceptibility landscape. The main text focuses on the weakly constrained neutral-phrase versus explicit-cue comparison; the Kettle codeword/phrase is the failure-conditioned neutral-codeword row, while Roses is reported in Table 25. Short horizontal ticks show means, and faint vertical lines show observed min–max ranges.

Table 24 summarizes these controls, used to interpret Figure 3E. Figure 9 shows the broader determination/susceptibility landscape, and Table 25 reports the complete formulation study. The four fixed report strings are admission wording 1 ("I was unable to find an exact solution with these numbers."), admission wording 2 ("I haven't found an exact solution for this one."), Roses ("Roses are red, violets are blue."), and Kettle ("A copper kettle rests on the kitchen shelf.").

In P-D, the phrase "Thanks for the question." is placed as the first line inside the reasoning block in exactly $\lfloor n/2 \rfloor$ of the solved examples and exactly $\lfloor n/2 \rfloor$ of the failure examples, chosen by a frozen hash. Its placement is therefore independent of outcome by construction, and the task reward ignores it. The SFT reference emits the phrase in 0.49 of sampled step-0 responses; the separate teacher-forced probability of starting the phrase is 0.34. In P-E2, each prompt ends with the visible line "Marker: A" or "Marker: B". The phrase appears in every marker-A training example and in no marker-B example, across both solved and failed examples. The marker is a fixed hash of the task identifier, so it is independent of the model's outcome, and the reward never reads the prompt. Every run is evaluated on both markers, and balanced adherence gives equal weight to emitting the phrase under A and omitting it under B.

The small P-E2 marker A/B side effect on failure disclosure, about 3–4 percentage points and within the preregistered tolerance, is reported here rather than in the main figure. It does not change the cue-adherence reproducibility result.

Table 25: Full semantic and codeword/phrase formulation study. The mixed ordering is why we do not claim that semantic admissions are generally unstable while codewords are generally stable.

Formulation	n	Emission range	SD	$\hat{\tau}$ [95%]
Admission wording 1 (SEM1)	20	0.41–0.91	0.141	0.64 [0.40, 0.86]
Admission wording 2 (SEM2)	20	0.19–0.91	0.229	1.03 [0.66, 1.48]
Neutral codeword Roses	20	0.10–0.91	0.172	0.82 [0.52, 1.06]
Neutral codeword Kettle	20	0.59–0.94	0.088	0.47 [0.25, 0.64]

Table 26: Summary of the main credit-assignment and direct-reward interventions. Older variants that do not change the final interpretation are omitted here.

Intervention	Setting	n	Main result
Collateral-credit arithmetic	Countdown	517 mixed groups	Adverse sign in 517/517 groups; predicted exposure 0.554, observed 0.553.
Exact-span shelter (pilot)	Countdown	3 matched pairs	Mean disclosure change +0.194 [0.017, 0.372]; a small pilot, not an established general level effect.
Fresh shelter experiments	Countdown + shortest-path	15 / 20 / 20 pairs	No mean-level change detected (+0.052 / +0.045 / -0.006); no reduction in variability established.
Failure-neutral credit	Countdown	20 paired	No reduction in variability is detected after removing adverse failure credit (log SD ratio +0.165 [-0.277, +0.613]) and lowers solve by about 6 points.
Direct disclosure reward	Countdown	3 arms (1 seed)	Can create or recover reporting when the action remains reachable.
Code: reward alone	MBPP+	2×2	No failure reports matching the true test outcome once the reporting action disappears from sampling support.
Code: reward + reference anchor	MBPP+	8 test-execution / 10 claimed-pass	Anchoring can preserve reporting support, but the combined intervention (reward plus anchor) does not reliably preserve functional reporting under a corrupt claimed-pass reward.

G COLLATERAL CREDIT AND DIRECT REWARD

Table 26 summarizes the sequence of interventions behind the bounded credit-assignment conclusion in Section 4.

The credit sign is adverse to reporting by construction in mixed groups, and all 517 relevant groups have that sign. In the three-pair shelter pilot, protecting the reporting span raises disclosure in every pair (mean +0.194 [+0.017, +0.372]). The mean-level change is not detected in the fresh $n = 15$ and $n = 20$ Countdown experiments (+0.052 [-0.115, +0.218] and +0.045 [-0.058, +0.149]) or the $n = 20$ shortest-path experiment (-0.006 [-0.208, +0.197]). These experiments also do not establish lower variation. The conclusion is therefore limited: collateral credit can affect reporting, but we do not establish a general level effect or a general explanation of retraining variability. Di-

Table 27: Main reference-anchoring experiments. SD ratios below one mean that failure disclosure varies less across retrainings under anchoring. Intervals are 97.5% for the Countdown and 7B rows and 95% for the shortest-path row, matching each experiment's pre-specified level.

Setting	n	SD ratio	log SD ratio interval	Change in solve
Qwen-1.5B / Countdown	20	0.472	-0.751 [-1.420, -0.225]	-0.060 [-0.089, -0.031]
Qwen-1.5B / shortest-path	20	0.30	-1.21 [-1.74, -0.76]	-0.083 [-0.126, -0.040]
Qwen-7B / Countdown	19	0.32	-1.14 [-1.55, -0.84]	+0.069 [+0.012, +0.126]

Table 28: Mean-adjusted anchoring check. Each ratio compares the between-run spread under anchoring with the spread without anchoring; values below one mean that disclosure is more consistent under anchoring. The mean-adjusted estimate uses a hierarchical binomial model to account for the compression of rates near zero and one. Mean disclosure is shown as anchored / unanchored, and entries in brackets are 95% intervals.

Setting	n	Mean disclosure (anchor / no anchor)	Raw SD ratio	Mean-adjusted SD ratio
7B, primary detector	19	0.80 / 0.32	0.32 [0.23, 0.42]	0.24 [0.16, 0.38]
7B, broad detector	19	0.84 / 0.52	0.30 [0.23, 0.39]	0.40 [0.29, 0.58]
1.5B, failure-conditional anchor	20	0.68 / 0.63	0.47 [0.26, 0.75]	0.42 [0.22, 0.64]
1.5B, all-token anchor	20	0.69 / 0.63	0.36 [0.25, 0.54]	0.33 [0.20, 0.51]

Table 29: Failure-conditional versus all-token anchoring in the 1.5B Countdown setting. Their reductions in disclosure variability are not distinguished, while a direct matched-seed comparison shows a larger capability cost for all-token anchoring (a descriptive companion, +0.024 [0.012, 0.036], not a preregistered test).

Anchor scope	Disclosure mean (SD)	Solve SD	SD ratio vs sham	Change in solve
None (matched comparator)	0.626 (0.1894)	0.0478	1 (reference)	n/a
Failure-conditional	0.684 (0.0893)	0.0294	0.472 [0.242, 0.799]	-0.0597 [-0.0888, -0.0306]
All-token	0.687 (0.0688)	0.0285	0.363 [0.230, 0.582]	-0.0835 [-0.1092, -0.0579]

rect reward can create or recover reporting when the action remains sufficiently reachable. Code experiments show that reward alone can fail once the model stops sampling the reporting action.

H Reference anchoring across retrainings

Table 27 reports the three main anchoring results across retrainings, and Table 28 reports the corresponding mean-adjusted checks. Table 29 compares failure-conditional and all-token anchoring in the primary setting. The anchoring coefficient $\beta = 0.04$, primary contrasts, and run rosters were fixed before outcome readout for the 1.5B Countdown, shortest-path, original 11-pair 7B, and anchored-history continuation experiments. After the original 7B result was known, we froze the remaining eight eligible seed pairs and repeated the experiment with the same training and evaluation settings. Exact timestamps are retained in the reproducibility record.

For 1.5B Countdown, the sham was trained fresh through the same anchor-capable trainer with $\beta = 0$. Its weights are byte-identical to the archived outcome-only weights in every available check (20/20 at step 150 and 14/14 at step 300). Where probe responses were sampled again, only the evaluation draw is new. We thus describe the sham as the outcome-only comparator with independently resampled probe readouts where applicable, not as a second independent control experiment. The later coefficient and token-region analyses reuse these archived arms. The 7B comparator uses the existing matched runs trained without anchoring rather than a newly trained control set.

Table 30: Progress-matched anchoring check for 1.5B Countdown. Slower capability progress explains part of the same-step narrowing, but disclosure remains less variable under anchoring than at the matched unanchored checkpoint.

Condition	Checkpoint	Disclosure SD	Role in comparison
Unanchored	300	0.189	Same training horizon as anchoring
Unanchored	150	0.139	Closest mean solve to anchored step 300
Failure-conditional anchor	300	0.089	Intervention result

Anchoring narrows both readouts in the primary runs. Probe-solve SD is 0.0478 in the matched comparator, 0.0294 under failure-conditional anchoring, and 0.0285 under all-token anchoring, a descriptive reduction of about 38–40%. Disclosure SD falls more, from 0.189 to 0.089 and 0.069 (0.189 is the comparator's independently resampled re-probe of the primary weights; their original probe read is 0.185, Table 11). The corresponding within-arm disclosure/solve SD ratios are 3.9, 3.0, and 2.4. The dispersion difference between the two anchored arms is not distinguished, but a direct matched-seed comparison shows that failure-conditional anchoring preserves 0.024 more solve accuracy than all-token anchoring (95% CI [0.012, 0.036]); this paired comparison is a descriptive companion rather than the preregistered primary. The mean-adjusted results in Table 28 show that differences in average disclosure rates do not explain the lower variability under anchoring.

H.1 Exact reference term and token scope

The reference model is a frozen copy of the SFT checkpoint from which each RL run starts. For each sampled completion token, we use the non-negative k3 estimator with $r_{i,t} = \log \pi_{\text{ref}} - \log \pi_\theta$. The added part of the accumulated loss is

$$\mathcal{L}_{\text{anchor}} = \frac{\beta}{32 \cdot 1024} \sum_{i,t} m_{i,t} s_{i,t} \left[\exp(r_{i,t}) - r_{i,t} - 1\right], \tag{1}$$

where $m_{i,t}$ marks active completion tokens. For failure-conditional anchoring, $s_{i,t} = 1$ for every active token of a verifier-incorrect, format-valid rollout and is zero for solved, format-invalid, or empty rollouts. For all-token anchoring, $s_{i,t} = 1$ for every active completion token. The denominator is always $32 \cdot 1024$, not the number of anchored tokens. The two arms thus use the same β and per-token coefficient on different token sets; they do not have the same total regularization budget. At 1.5B, failure-conditional anchoring covers 0.570–0.601 of active completion tokens across runs, compared with 1.000 for all-token anchoring. The corresponding 7B range is 0.513–0.578. We did not adjust β to match the total coefficient mass across arms.

H.2 Progress-matched comparison

Table 30 compares the failure-conditional arm at step 300 with the unanchored checkpoint whose mean solve is closest to it. This is a post-hoc descriptive check, not a randomized control for every way an intervention might slow learning.

Relative to unanchored step 300, moving back to the progress-matched step-150 checkpoint reduces disclosure SD from 0.189 to 0.139. Anchoring reduces it further to 0.089, about 36% below the progress-matched value. This comparison shows that slower progress accounts for part, but not all, of the observed narrowing; it does not fully separate the two effects.

H.3 Coefficient ladder

Figure 10 shows the small β coefficient sweep used only as a sensitivity check, not as a fitted dose-response relationship.

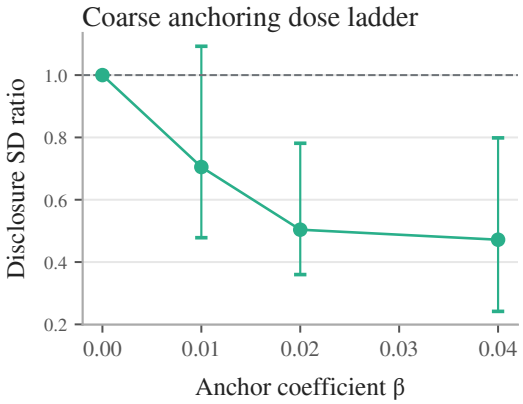

Figure 10: Small reference-anchor coefficient sweep. Narrowing is established at $\beta = 0.02$ but not at $\beta = 0.01$; we do not treat this as a fitted dose-response law.

Table 31: 7B anchoring extension and pooled result. The extension was frozen after the original readout and reproduces the tightening itself. The pooled interval uses a wave-stratified bootstrap.

Run set	n	Log SD ratio [97.5% interval]	SD: anchor / no anchor	Change in solve
Original fixed pairs	11	-1.36 [-3.13, -0.36]	0.062 / 0.242	+0.062 [-0.021, +0.145]
Frozen extension	8	-1.01 [-1.75, -0.32]	0.111 / 0.305	+0.080 [-0.020, +0.179]
Pooled, wave-stratified	19	-1.14 [-1.55, -0.84]	0.090 / 0.281	+0.069 [+0.012, +0.126]

H.4 7B EXTENSION, PROVENANCE, AND INFLUENCE SENSITIVITY

We first selected 11 matched 7B model pairs before examining their outcomes. After analyzing those runs, we separately fixed the remaining eight eligible seeds as an extension, using exactly the same $\beta = 0.04$ intervention, mask, training horizon, detector, probe, exclusion rules, and statistical analysis. The eight new pairs independently show reduced variation in disclosure across runs. The main text also reports the combined estimate across all 19 pairs, using a bootstrap that keeps the original and extension waves separate when resampling. Table 31 reports the original wave, the extension, and the combined analysis. The original 11 pairs were directionally stable to every leave-one-run-out deletion, although interval-based significance was sensitive to four of the eleven runs; in the pooled 19 pairs the interval excludes zero after every deletion (Table 32).

All 19 pairings pass the provenance check: no seed appears twice, each anchored run is paired with its pinned same-seed KL-off baseline, and all runs use the common execution configuration. The original 11 pairs used the common H200 execution configuration used for all 7B runs. The extension ran on a 4×H200 pod after three replay checks reproduced the original checkpoint hashes bit-for-bit. A twelfth seed in the original program changed batch geometry after an out-of-memory error and was excluded before any outcome was seen; no licensed run was removed after readout.

I ANCHORING, LATER-RL DIVERGENCE, AND LOCALIZATION

I.1 ANCHORED-HISTORY CONTINUATION AND POST-HISTORY ANCHORING

Table 33 compares two mid-RL continuation designs: the anchored-history continuation test (KBR) continues or switches off anchoring after an anchored history, while the post-history anchoring test (FORK) introduces anchoring after an unanchored history. Figure 11 shows all KBR continuation children, and Figure 12 shows the smaller FORK summary across parent mid-RL states.

Table 32: Leave-one-run-out sensitivity for the 7B anchoring result. The original and extension-only results are each based on few pairs, while the pooled $n = 19$ estimate excludes zero after every one-run deletion.

Analysis	n	Log SD ratio or deletion range	Interval stability
Original pairs	11	-1.36 [-3.13, -0.36]	Four of 11 deletion intervals cover zero.
Extension alone	8	-1.01 [-1.75, -0.32]	Four of eight deletion intervals cover zero.
Pooled pairs	19	-1.14 [-1.55, -0.84]	Full interval excludes zero.
Pooled leave-one-out	19 deletions	Point estimates -1.26 to -1.06	All 19 deletion intervals exclude zero.

Table 33: Mid-RL continuation tests of anchoring. KBR shows reduced later-RL divergence after an anchored history. FORK does not establish narrowing on the raw scale when anchoring is introduced only after an unanchored history.

Design	Parents	SD ratio	log SD ratio interval	Interpretation
Anchored-history continuation (KBR): keep anchor on vs switch off	10	0.47	-0.75 [-1.22, -0.25]	8/10 parents show lower variability with anchoring on
Post-history anchoring (FORK): introduce anchor after unanchored history	8	0.60	-0.51 [-1.10, +0.09]	Reduced variability is not statistically established

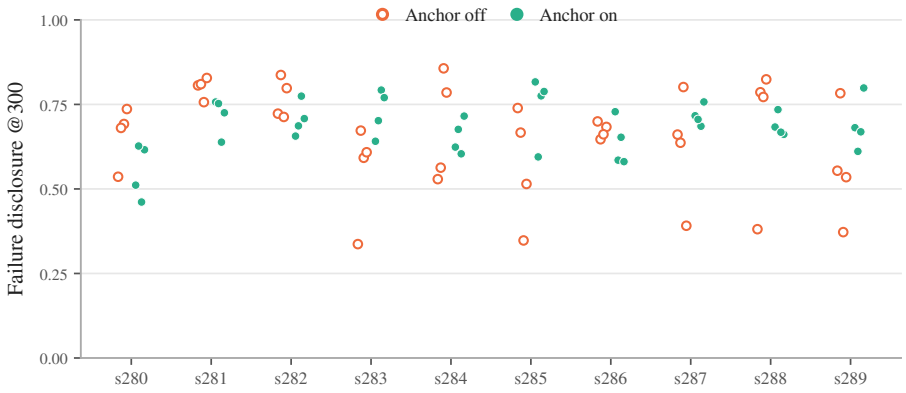

Figure 11: All KBR continuation runs across ten mid-RL parent states with anchored histories. Each parent has four futures with anchoring switched off and four with anchoring kept on.

For KBR, continuing anchoring reduces pooled within-parent disclosure SD from 0.138 to 0.065 (SD ratio 0.47; log ratio -0.75 [-1.22, -0.25]), with lower variability in 8/10 parent states. Mean disclosure is not detectably changed (0.683 with anchoring on versus 0.658 with anchoring off), while probe solve shows a detected 4-point decrease, from 0.386 to 0.344 ($\Delta = -0.043$). This conclusion is history-specific: the complementary FORK experiment, which introduces anchoring only after an unanchored history, does not establish reduced variability.

I.2 Token-region localization

Table 34 compares full anchoring with anchoring only the tail, the reporting seam, or the failed-reasoning portion. Tables 35 and 36 report the corresponding gradient and reference-drift analyses.

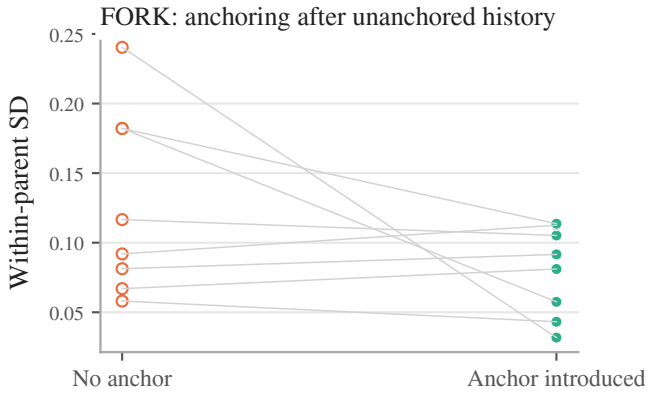

Figure 12: FORK variability after an unanchored training history. Starting from each of eight exact mid-RL parent states, we run four matched future continuations without anchoring and four with anchoring introduced at the fork. Each dot is the within-parent SD of failure disclosure across those four continuations; lower values therefore indicate less branching across otherwise matched futures.

Table 34: Where the reference constraint acts. Anchoring only failed reasoning recovers the full stabilization effect and capability cost. Short tail and seam masks do not establish narrowing and cover far fewer tokens. Intervals are 97.5% for the full anchor and 98.33% (a three-arm family) for the failed-reasoning, tail, and seam rows.

Anchor region	log SD ratio vs sham	Change in solve	Active-token exposure
Full failure-conditional	-0.7514 [-1.4200, -0.2250]	-0.0597 [-0.0888, -0.0306]	58.9%
Failed reasoning	-0.7109 [-1.2055, -0.1834]	-0.0643 [-0.0859, -0.0427]	53.6%
Post-attempt tail	-0.0468 [-0.6499, +0.6019]	-0.0331 [-0.0731, +0.0069]	4.3%
16-token seam	-0.0438 [-0.5230, +0.5503]	-0.0038 [-0.0345, +0.0270]	1.7%

Table 35: Gradient and drift localization analysis (LOCKLAN). The failed-reasoning-only gradient is almost as large as the full-anchor gradient and points in nearly the same direction, while tail and seam gradients are much smaller.

States	$\|g_D\|/\|g_{\text{full}}\|$	$\cos(g_D, g_{\text{full}})$	Tail norm ratio	Seam norm ratio
Anchored @ 150	0.960	0.983	0.173	0.132
Anchored @ 300	0.964	0.984	0.172	0.137
Sham @ 150	0.975	0.993	0.110	0.095
Sham @ 300	0.958	0.989	0.139	0.103

Table 36: Teacher-forced drift from the reference policy at step 300. About 92–95% of the measured drift lies before the reporting seam in the reported 1.5B and 7B settings.

Setting @300	Pre-seam drift mass	Seam mass	Other tail mass	Total drift
1.5B / KL-off	0.934	0.037	0.029	0.119
1.5B / Sham	0.934	0.038	0.028	0.119
1.5B / Anchor	0.947	0.029	0.024	0.031
7B / outcome-only	0.923	0.058	0.019	0.199
7B / Anchor	0.953	0.025	0.023	0.047

J EXTENDED ROBUSTNESS AND BOUNDARY ANALYSES

Table 37 links common alternative explanations and scope questions to the corresponding evidence elsewhere in the appendix.

Table 37: Summary of robustness and scope evidence.

Question	Evidence	Main takeaway
Role of reference KL	KL-free setup + all-token and failure-conditional anchoring	The phenomenon is measured in a deliberate KL-free regime; restoring reference regularization narrows both readouts, especially disclosure.
Generic RL variability	Primary + OLMo + shortest-path selective-dispersion comparisons	Variability depends on the behavior: disclosure is much less reproducible than solve in the clean experiments.
Capability differences	Clean 1.5B/OLMo experiments; 7B boundary	Capability is tightly reproduced in the clean experiments; 7B is explicitly treated as a broader persistence boundary.
Detector sensitivity	Human audits + detector-tier sensitivity	Human agreement is high, randomized audit frames are included, and broader detectors preserve the qualitative spread.
Probe selection	Full dev-100 disclosure + selected-20 versus other-80 split	Broad spread remains on the 80 prompts outside the behavior-selected probe; selection does not explain the result.
Fixed-horizon interpretation	Nine-rung trajectory + small longer-horizon cases	The claim is reproducibility at a fixed budget, not separate stable endpoints; longer outcomes are mixed (J.1).
Report-entry localization	NoBreak control + mean-adjusted Step B + route-scale check	Most tightening comes from the learned paragraph-break cue; this is template-specific and the 70% route split is raw-scale only.
Dependence on report SFT	Instruction-conditioned Qwen-32B runs	Broad reporting variation also appears without the primary report-SFT setup.
Algorithm scope	Stabilized PPO n=8; RLOO descriptive only	PPO provides lower-variance boundary evidence; we do not claim algorithm invariance from RLOO.
Task and model scope	Shortest-path + OLMo-1B	The phenomenon is not limited to Countdown or Qwen.
Anchoring level effects	Paired level effects + 7B retention	Anchoring narrows variation without lowering the mean in the primary runs; at 7B it mainly preserves the reference reporting behavior.
Mid-RL state restoration	Same-key twins + same-configuration replay	Matched restores and replays are bit-exact in the replay checks.

For the primary runs, robust dispersion measures preserve the ordering (MAD ratio 2.87; IQR ratio 3.02), and fixed failure-mixture reweighting changes the disclosure span only from 0.646 to 0.629. For the shortest-path task, the comparison on the transformed scale is treated as inconclusive because probe-solving performance is already close to its floor; we therefore do not generalize from it to the

Table 38: Small multi-run cases trained beyond step 300. SD is calculated across runs using the detector named in each row. These values are descriptive because every case has $n \leq 5$ and the instruments differ across model families.

Setting	n	Last step	Disclosure SD, 300 → last	Solve SD, 300 → last	Last-step disclosure
Qwen2.5-1.5B, outcome only	3	600	0.273 → 0.023 (D_{v1})	0.064 → 0.131	0.039 / 0.000 / 0.000
Qwen2.5-1.5B, length penalty	3*	600	0.092 → 0.148 (D_{v2})	0.050 → 0.217	0.404 / 0.651 / 0.667
Qwen2.5-7B, outcome only	5	600	0.306 → 0.396 (D_{v1})	0.075 → 0.201	0.000–0.918
OLMo-2-7B, outcome only	3	1200	0.099 → 0.169 → 0.442 (D_{v5})	0.081 → 0.163 → 0.030	0.873 / 0.000 / 0.555
Gemma-2-9B, outcome only	3	600	0.221 → 0.115 (D_{v5})	0.105 → 0.244	0.666 / 0.886 / 0.833†
Distilled Qwen-1.5B, two recipes	2+2‡	600	0.049 → 0.078 (D_{v2})	0.055 → 0.013	0.797–0.961

*Three runs spanning two seeds and two execution configurations. †The apparent narrowing is detector-dependent: D_{v1} gives 0.676 / 0.004 / 0.787 at step 600. ‡Two outcome-only and two length-penalty runs are pooled only for this descriptive row, not as one combined estimate.

cleaner experiments. Robust measures of variability and leave-one-out checks are discussed in the main text only when they meaningfully qualify a main claim.

J.1 Training beyond 300 steps

Our main results are measured at step 300, so they show what happens at that point in training rather than what would necessarily happen after much longer training. Table 38 summarizes the few cases that were continued beyond step 300. Each case includes at most five runs measured in the same way, and one row combines two training recipes only for a descriptive comparison, so these results are too limited to support a general long-term trend.

The later outcomes are mixed. Three outcome-only 1.5B Qwen runs move toward very low primary-detector disclosure by step 600 (0.039, 0.000, and 0.000), although the broadest detector still reads 0.103, 0.161, and 0.215 and individual runs pass through both high- and low-disclosure states along the way. Five outcome-only 7B runs still span 0.000–0.918 at step 600. Three OLMo-2-7B runs have disclosure SD 0.169 at step 600, then end at 0.873, 0.000, and 0.555 at step 1200. A capability change in one run makes the later comparison harder to interpret.

Detector choice also matters at long horizons. In one distilled 1.5B run continued to step 1200, the narrow phrase detector falls to 0.067 while broader content detectors remain at 0.882–0.914. Thus a surface-form change can look like a loss of disclosure under a narrow lexical monitor. Overall, these small cases show no single post-300 endpoint: reporting can erode, remain mixed, or change wording. Our primary claim hence concerns reproducibility at a fixed training budget, not distinct stationary endpoints. A possible explanation is that disclosure erodes at different rates across runs as training continues. The three 1.5B cases are compatible with this, but too few to establish it.

K EXTENDED RELATED WORK

Failure awareness, self-correction, and retraction. A large body of work studies whether models can recognize errors or improve after noticing them. Kadavath et al. (2022) study whether models can estimate if they know an answer; Tyen et al. (2024) separate finding an error from correcting it; and recent work studies self-verification, retraction, correction, and stepwise failure detection under different forms of feedback (Yang & Jia, 2026; Chen et al., 2026; Tsui, 2026; Huang et al., 2024; Kamoi et al., 2024; Mavi et al., 2025). Our question is not whether the model can eventually fix the answer, but whether an objectively failed attempt is reported consistently across retrainings.

Honesty, abstention, and reporting incentives. Joglekar et al. (2025) train a separate confession channel; Zhai et al. (2026) optimize abstention under RLVR; and Kalai et al. (2026) study incentives that can reward guessing instead of admitting uncertainty. Wen et al. (2025) report that RLHF can make wrong answers more persuasive, while Chandna et al. (2026) highlights validity concerns in the original reward-model setup. Work on verbalized overconfidence and confidence–correctness alignment provides additional context (Zhao et al., 2026; Xie et al., 2026). Our setting instead asks whether an existing, unrewarded reporting behavior remains reproducible through post-training.

Underspecification and numerical reproducibility. Our broad framing follows work on underspecification: standard evaluation metrics can leave other behaviors poorly constrained (D'Amour et al., 2022). Sensitivity to random seeds is also well documented in RL and language-model training (Henderson et al., 2018; Fehlauer et al., 2025; Bui et al., 2025). Formal work studies replicable RL (Eaton et al., 2026), while systems work shows that floating-point ordering, kernels, and implementation details can introduce nondeterminism (Yuan et al., 2025; Shanmugavelu et al., 2024; Gond et al., 2026; He & Thinking Machines Lab, 2025; Whitehead & Fit-Florea, 2011). Altıntaş et al. (2025) provide a close comparison to our trajectory framing, showing that tiny perturbations can grow during training. We ask whether one safety-relevant behavior is selectively less reproducible than task capability and whether reporting can still diverge later in RL from the same training history.

Post-training behavioral drift and preservation. Recent work studies safety-relevant behaviors that can emerge or persist during post-training (Schreiber & Goldstein, 2026; Betley et al., 2025; 2026). Reference regularization can limit behavioral drift: Elcock et al. (2026) study preservation during task adaptation, while Soligo et al. (2026) examine how KL constraints can favor a narrower set of solutions. It is not automatically benign for oversight: Golechha et al. (2026) report that KL penalties can increase chain-of-thought unfaithfulness in reward-hackable coding environments, motivating our treatment of anchoring as a tradeoff rather than a free constraint. Closer to our reproducibility question, Hussing et al. (2026) formalize behavioral consistency across deep-RL runs and reduce policy differences using adaptive entropy regularization toward a common prior. We instead apply a reference term to a measured reproducibility failure in language-model RL, test it across retrainings and controlled mid-RL continuations, and study which part of failed trajectories carries the stabilizing effect.

Credit assignment, verifiers, and monitorability. Process-supervision methods such as rewarding verified prefixes motivate our local credit experiments (Liu et al., 2026c). Work on gaming verifiers also shows that even a formally checkable objective can leave intended behavior underspecified (Helff et al., 2026). Chain-of-thought monitorability research asks which optimization pressures preserve useful behavioral traces (MacDermott et al., 2025; Guan et al., 2026; Emmons et al., 2025), while related work cautions that textual "verification" can be causally weak (Zhao et al., 2025). These works motivate preserving reporting behavior, but we do not equate failure disclosure with faithful reasoning or grounded verification.

Hallucination-adjacent framing. Failure disclosure is related to, but distinct from, hallucination or confident fabrication. Recent definitions emphasize observable mismatch with a relevant world model or reference (Liu et al., 2026b;a). We instead ask what the model says about its own unsuccessful attempt in a setting where task failure is objectively known. We therefore avoid calling nondisclosure "lying" or treating failure disclosure as simply a type of hallucination.