WHAT YOU DON'T CHECK, MODELS DON'T FIX: FALSE COMPLETION UNDER PARTIAL VERIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Iterative correction loops revise LLM outputs until their checks pass, and self-improving pipelines favor changes that score better on theirs. However, the checks rarely cover every way an output can be wrong, so an output accepted as correct may well contain defects that the checks missed. We ask what happens to these defects. We study a correction loop on three task domains (music-as-code audio, SVG repair, and API workflows) with eight proprietary and open-weight models. Each task has eight deterministic checks. All eight checks are scored in every round, but the model sees only some of them (the exposed checks; the rest are hidden), and the loop stops when the exposed checks pass. Because the hidden checks are also scored, each run shows which defects the loop repairs and which it leaves behind when it stops. First, models fix what they are shown: exposed defects are repaired far more often than hidden ones, and at low coverage most of the loop's stops leave a hidden defect, usually before any revision. Second, stopping too early is not the main reason these defects remain: extra rounds recover little, and telling the model that its output is still wrong helps, but far less than naming the failing checks. Third, models can compensate only partly: reasoning reduces the effect of coverage without removing it, and letting models choose which checks they see does not detectably reduce the remaining defects. What you don't check, models don't fix: a caution for any correction loop trusting partial verification.

1 INTRODUCTION

Correction loops improve LLM-generated artifacts by alternating verification and revision: tests, verifiers, judges, or environments report failures, the model revises, and the loop stops when the reported checks pass. Self-improving pipelines apply the same step to an agent's own code or training recipe and favor changes that score better on checks that are partial by design, such as staged benchmark subsets (Zhang et al., 2026b) or execution feedback that is "reliable but incomplete" (Chen et al., 2026a). Such loops are usually judged by how well the model uses feedback; we study a different variable, the *coverage* of the verification: how many of the ways an artifact can be wrong are checked and reported. In practice, verification is selective: test suites cover a subset of behaviors, judges sample properties, agentic evaluators choose which checks to run (Chen et al., 2026b), and environments reveal only part of their state. If an artifact can be wrong in eight checkable ways but the loop tests only two, an "all-clear" may mean correctness, or only that those two pass while others still fail, because the first attempt already passed them or because revision was steered toward them.

Existing work has studied the two extremes of coverage. With *no* external signal, models largely fail to correct themselves on math and question-answering tasks (Huang et al., 2024); with *sound and complete* verification, studies ask how models use it and how detailed its critiques should be (Stechly et al., 2025). Work in between studies false success claims (Advani, 2026) and content-free failure notices (Verma, 2026), and finds that feedback helps only as far as its checking tools observe (Li & Shi, 2026). The closest precedent predates LLMs: patches from search-based program-repair tools overfit more when their test suite covers less (Smith et al., 2015). None of these studies fixes a set of checks and varies how many of them both the model and the stopping rule see while still scoring every check, which lets us compare exposed and hidden defects within the same run.

We therefore run a correction loop in which we control how many of the checks the model sees, across three task domains and eight proprietary and open-weight models (Fig. 1). Every task has

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.

eight deterministic checks. All eight checks are scored in every round, but the model sees only K of them, and the loop stops when those K pass. Within each run, we can thus compare the defects the model was told about with those it was not, and we use this comparison to answer three questions: what partial verification does to repair and stopping (Sec. 5), why the loop stops with defects left (Sec. 6), and whether models can compensate (Sec. 7).

Contributions.

- **A controlled measure of what partial verification leaves behind.** The loop scores every check but reports only some, and every coverage level starts from the same first draft, so the repair of reported and unreported defects can be compared within the same runs (Sec. 3).
- **Repair locality and false completion.** Models repair the defects they are shown far more often than those they are not, in all 16 audio and SVG model–domain cells. At low coverage, most of the loop's stops leave a hidden defect, and three quarters of these stops happen before any revision (Sec. 5).
- **Missing information, not missing rounds.** Extra rounds recover little. Telling the model that its output is not fully correct helps slightly beyond offering another round, but naming the failing checks helps far more (Sec. 6).
- **Limited compensation from the model.** Reasoning roughly halves the effect of coverage without removing it, helping both the first draft and the repair. Letting models choose their checks changes which checks they see but not how many defects remain (Sec. 7).

2 Related work

Self-correction, self-improvement, and external signals. Without new external information, self-correction largely fails on math and question-answering tasks (Huang et al., 2024); it helps where the model can detect its own flaws (Madaan et al., 2023) or after training (Kumar et al., 2025), and multi-round instruction compliance remains brittle (Wang et al., 2026). Self-improving pipelines score changes to the agent or its training algorithm on benchmark subsets or proxy metrics (Zhang et al., 2026b; Chi et al., 2026; Chen et al., 2026a). Reflection and search frameworks (Shinn et al., 2023; Zhou et al., 2024) vary the revision policy (here a single chain); their interaction with coverage is untested. Closest in spirit, Stechly et al. (2025) let a sound verifier decide acceptance and vary how much of its critique the model sees (that the answer is wrong, the first error, or every error): the amount makes little difference, and even asking the same question again, with no critique, until the verifier accepts keeps most of the improvement, which therefore comes mostly from the sound verifier itself rather than from what the critique says. In our loop, by contrast, the stop decision sees only the checks the model sees, so it can accept an artifact that is still wrong, and we vary how many checks the model sees, down to none. In our experiments, feedback content does matter: a "not fully correct" message helps far less than naming the failing checks (Sec. 6).

Hidden tests and specification gaming. Execution feedback improves code repair once models are trained with reinforcement learning to use it (Gehring et al., 2025); yet patch overfitting to visible tests is a long-standing failure of program repair (Smith et al., 2015). Much larger test suites reclassify a substantial share of generated code previously scored correct (Liu et al., 2023), and on SWE-bench, patches that pass auto-generated tests frequently fail hidden reference tests, with iterative refinement able to worsen the overfitting (Ahmed et al., 2026). Reward hacking is the same failure when a policy is optimized against a proxy reward (Skalse et al., 2022), and Arimbur (2026) compares resampling with repair from execution error messages, without varying how much verification is disclosed. We measure the exposed-versus-hidden gap within trajectories, under fixed random masks, scoring every check.

False success and the content of feedback. Agents frequently claim success that the environment contradicts (Advani, 2026; Kumar et al., 2026). Completion gates forbid a "done" claim unless a check has run and passed (Deng, 2026); a tool for reporting a defective test, plus a rule against exploiting it, cuts agents' exploitation of such tests from 23.6% to 5.3% (Gomez, 2026); and hidden tests derived from code review reject a third of the repairs that pass the functional tests (He et al., 2026). None of these varies how many check results the agent sees. Our all-clear stop is such

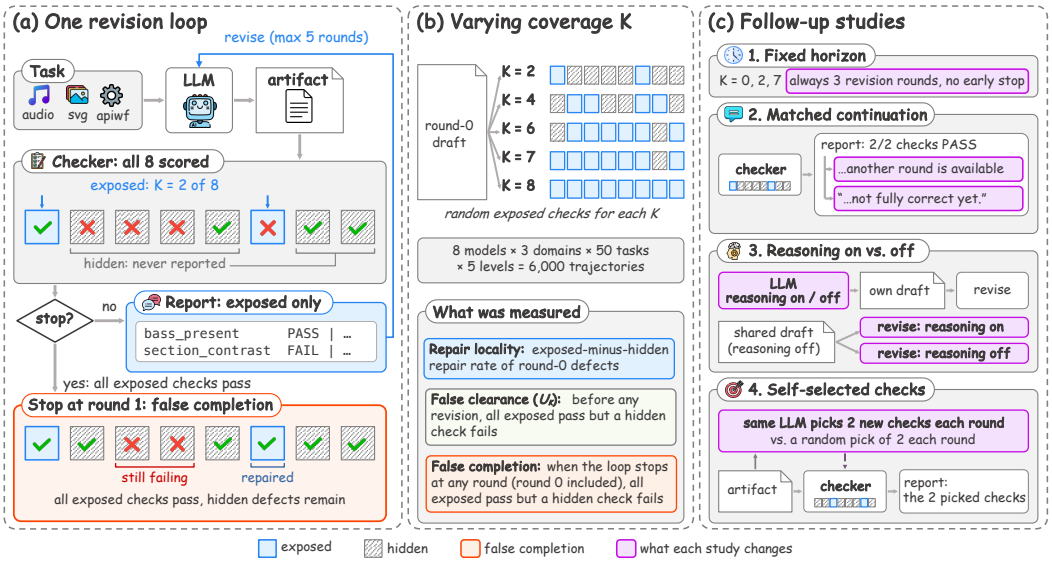

Figure 1: **Study design.** (a) All eight checks are scored every round; only the K exposed ones are reported. The loop stops when all exposed checks pass, including at round 0, or after five revision rounds. The example (an audio task at K=2; Appendix F) stops after one revision with two hidden failures, a false completion. (b) Coverage varies over $K \in \{2, 4, 6, 7, 8\}$, sharing one round-0 draft per (task, model), with masks drawn per (task, K) and fixed across models. (c) Follow-ups: the three-domain fixed-horizon control (Sec. 6.1), the matched continuation control, whose two branches append the neutral or the "not fully correct" sentence to the same report (Sec. 6.2), reasoning at generation and repair (Sec. 7.1), and self-selected checks (Sec. 7.2); full design: Fig. 5.

a gate with partial coverage, and every false completion passes it; what remains to design is the report's content. The closest designs vary that content: for small code models a content-free failure notice works as well as real execution feedback (Verma, 2026); for one repair-agent setup, reporting whether the agent's own passing test also passes on the unfixed code (if so, it cannot detect the bug) reduces how often the agent finishes with only such tests as evidence more than a generic reminder of matched form, which alone gives about a third of that improvement (Xu & Wu, 2026); and counterexample feedback dominates error-only feedback on regex synthesis (Badrinarayan & Parthasarathy, 2026). None varies how many of a fixed set of checks are reported.

Allocating and evaluating verification. Budgeted-verification allocators decide when and what to verify (Patel, 2026; Tang & Zhan, 2026; Dip et al., 2026). Judge-based evaluation validates AI judges (Zheng et al., 2023), including judges that choose which checks to run per case (Chen et al., 2026b); process- versus outcome-verification work asks where supervision attaches (Lightman et al., 2024); and interaction-scaling evaluation finds grounded feedback limited to what its checking tools observe (Li & Shi, 2026). Even frontier models locate the failure-inducing step in traces from hand-crafted multi-agent systems less than 20% of the time (Zhang et al., 2026a), much as API-workflow failures resist repair (Sec. 5.3). These works study the verifier; we study the verified model: how many checks it is shown and whether it picks them.

3 Method: A Correction Loop with Partial Verification

We keep the correction loop fixed and change one thing: how many of its checks the model is told about. Formally, each task has eight deterministic checks $C = \{c_1, \ldots, c_8\}$, and $c(x) \in \{0, 1\}$ records whether artifact x passes check c. Starting from a draft x_0, in round t the model sees a report on the exposed checks $E \subseteq C$ with $|E| = K$ and revises x_{t-1} into x_t. The loop stops at $\tau = \min\{t \leq T : c(x_t) = 1 \text{ for all } c \in E\}$, or at the horizon T if no such round exists. All eight checks are scored at every round, so for each round-0 defect ($c(x_0) = 0$) we observe whether it is repaired ($c(x_\tau) = 1$), for exposed checks $c \in E$ and hidden checks $c \in H = C \setminus E$ alike. Below

we describe the loop (Fig. 1a), partial verification and how we keep the comparisons fair (Fig. 1b), what we measure, and the variants used in Secs. 6 and 7 (Fig. 1c); Sec. 4 gives the concrete tasks, models, and settings.

The loop. A correction loop alternates checking and revision (Fig. 1a). The model first writes an artifact for a task, the *round-0 draft*. Deterministic *checks*, programs that each test one property of the artifact and return pass or fail, score it; a failing check is a *defect*. The model then receives a *check report* and revises the artifact. The loop stops when all reported checks pass (an *all-clear stop*) or after a fixed number of revision rounds (the *horizon*). One run of the loop for one task, model, and coverage level is a *trajectory*.

Partial verification. Every task has eight checks, and all eight checks are scored in every round, but the report shows only K of them, the *exposed* checks; the other $8-K$ are *hidden*, and K is the *coverage level*. Because hidden checks are still scored, we see what the loop leaves behind: when all exposed checks pass while a hidden check fails, the artifact is in a *false-clear state*, and the all-clear stop it triggers is a *false completion*. Formally, x is false-clear if $c(x) = 1$ for all $c \in E$ and $c(x) = 0$ for some $c \in H$, and a trajectory ends in false completion if x_τ is false-clear. Two choices keep the comparisons fair. Within a trajectory, exposed and hidden defects share the same model, task, draft, and rounds. Across coverage levels, every K starts from the same round-0 draft, and the exposed checks (the mask) are drawn at random per (task, K) and fixed across models (Fig. 1b).

Measures. **Repair locality (P1)** is the repair rate of exposed round-0 defects minus that of hidden ones in the same trajectories, where a defect is *repaired* if its check passes when the loop stops. In symbols, $\mathrm{P1} = r_E - r_H$, where r_S is the share of round-0 defects $c \in S$ with $c(x_\tau) = 1$. **False clearance (P2)**, $U_K = \Pr[x_0 \text{ is false-clear}]$, is the probability that the round-0 draft is already in a false-clear state, so the loop stops before any revision. We also report the *residual defect count* $D = |\{c \in C : c(x_\tau) = 0\}|$ when the loop stops and whether the artifact is *fully correct* ($D = 0$).

Variants of the loop. To ask why false completions occur and whether models can compensate, we change one element of the loop at a time (Fig. 1c): the stopping rule (a fixed horizon, Sec. 6.1), the report (a message that the artifact is not fully correct, against a neutral offer of another round, Sec. 6.2), the model's reasoning (Sec. 7.1), and who chooses the exposed checks (Sec. 7.2).

4 Experimental setup

Domains, tasks, and checks. We use three executable artifact domains with external ground truth (Appendix Table 2): **audio**, **svg**, and **apiwf**. Each domain has 50 tasks built from real artifacts, so every task is satisfiable (Appendix A). Each task's eight checks are programs rather than learned judges, and each failing check returns a structured counterexample (examples: Appendix F). In audio tasks, the model composes a new piece in our music-as-code DSL from a given specification. The specification and six of the eight thresholds come from a public-domain PDMX musical score (Long et al., 2025); converted to the DSL, this score passes all eight checks. The other two checks (`no_clipping` and `bass_present`) use the same thresholds in every task. In svg tasks, the model repairs a given graphic. To mimic real-world repair, each graphic starts from an SVGenius graphic (Chen et al., 2025) that passes all eight checks and receives one edit for each of four to six targeted checks, so the other two to four checks pass from the start (Appendix A). In apiwf tasks, the model writes a program from the API manual, and the program runs in the official AppWorld sandbox (Trivedi et al., 2024). Each apiwf task is a new instance of one of ten AppWorld task families: it keeps the family's world and instruction wording, changes one instruction parameter, and has a reference program that passes AppWorld's own evaluation and our eight checks.

Reports, masks, and stopping. The model cannot run any check itself: many audio and SVG checks require rendering, and apiwf programs run only after submission, not interactively as in standard AppWorld. Each round's check report uses a fixed template: it lists the verdicts of the K exposed checks with one counterexample for each failing one, and a fixed directive to fix the reported failures follows. Each (task, K) mask is drawn once and fixed for all models, and the draws are balanced so that every check is exposed about equally often. The horizon is five revision rounds,

and an all-clear stop before the horizon is *early stopping*. Raising K also makes the report longer and delays the stop; each exposed check adds one line, and the report is 1.2–3.2% of the prompt (Appendix E). Within a trajectory, exposed and hidden checks share both effects. Across K, the three-domain fixed-horizon control (Sec. 6.1) removes early stopping, and padding the $K{=}2$ report to $K{=}8$ length with content-free lines leaves hidden-defect repair unchanged (Appendix B).

Models and main grid. The main grid crosses $K \in \{2, 4, 6, 7, 8\}$ with 8 models, 3 domains, and 50 tasks, for 6,000 trajectories, with reasoning disabled and `max_tokens` 16,384. Five models are proprietary: GPT-5.4 (OpenAI, 2026), Claude Sonnet 5 (Sonnet-5 below; Anthropic, 2026), Claude Haiku 4.5 (Haiku-4.5; Anthropic, 2025), Gemini 3.5 Flash (Gemini-3.5; Google DeepMind, 2026), and Qwen3.8-Max (Qwen Team, 2026b); three are open-weight: DeepSeek-V4-Pro (DeepSeek-AI, 2026), Qwen3.5-27B (Qwen Team, 2026a), and Gemma-4-31B (Gemma Team, 2026). Appendix A lists each model's API identifier and access route. Calls set temperature 0, top-p 1, and seed 0 where the interface allows; the effective configuration differs by interface (Appendix Table 6).

Statistics. Checks differ in scope: in apiwf, `completeness` alone runs AppWorld's full task-outcome assertion suite, so $K/8$ is the exposed share of a domain's own checks and is compared only within a domain. P1 is computed within each partial coverage level $K \in \{2, 4, 6, 7\}$ and averaged across levels with defect-count weights. We also report *conditional false completion*, the share of all-clear stops at which a hidden check still fails. P1 and P2 are estimated with a task-level bootstrap ($B{=}10{,}000$; a resampled task carries all of its conditions), and P1 is tested per cell with Holm correction across all 24 cells.

5 RQ1: What happens under partial verification?

5.1 Repair locality

Models repair the defects they are shown and largely leave the rest. In audio and SVG, exposed round-0 defects are repaired far more often than hidden ones in the same trajectories, and this repair locality (P1) is significant in all 16 model–domain cells (Fig. 2a). At the extreme, Sonnet-5 on SVG repairs 100% of its exposed round-0 defects but only 2% of its hidden ones at $K{=}2$. API workflows show the effect only weakly (Sec. 5.3).

The gap is not explained by early stopping or by exposed checks being inherently easier to fix. Without early stopping the gap remains: when the loop runs exactly three revision rounds at $K{=}2$, exposed defects are still repaired more often than hidden ones for all eight models in audio and SVG, and in API workflows pooled across models (Appendix Table 11); Sec. 6.1 tests how much stopping early matters for the defects that remain. Exposed checks are also not inherently easier to repair: because masks differ across tasks, the same check is exposed in some tasks and hidden in others, and overall, at $K{=}2$ it is repaired more often when exposed in all 16 audio and SVG cells (Appendix B). The comparison can use a check only if, within a cell, it fails in at least one round-0 draft where it is exposed and in at least one where it is hidden. In API workflows, 80% of round-0 defects fall in `completeness`, which meets this condition in every cell but is almost never repaired, whether exposed or hidden; the other checks mostly pass in the first draft, so the comparison is not informative there (Appendix B).

5.2 False completion under low coverage

At low coverage, most of the loop's stops are false completions. Because the stopping rule sees only the exposed checks, the loop can stop while a hidden check still fails. At $K{=}2$, more than half of the all-clear stops are false completions in 23 of the 24 model–domain cells; the exception is Gemma-4-31B on audio, at 40% (Fig. 2d).

Most of this is decided before any revision. Of the 888 false completions at $K{=}2$, 668 (three quarters) happen at round 0: the first draft already passes the two exposed checks while a hidden check fails, so the loop stops without revising (Fig. 2e). We call these drafts *falsely cleared*; their share is the false clearance U_2. The other 220 follow revisions that fixed the failing exposed checks while a hidden check still failed, which is where repair locality enters. False clearance is higher at $K{=}2$ than at $K{=}7$ in all 24 cells (Fig. 2b; Appendix Table 8).

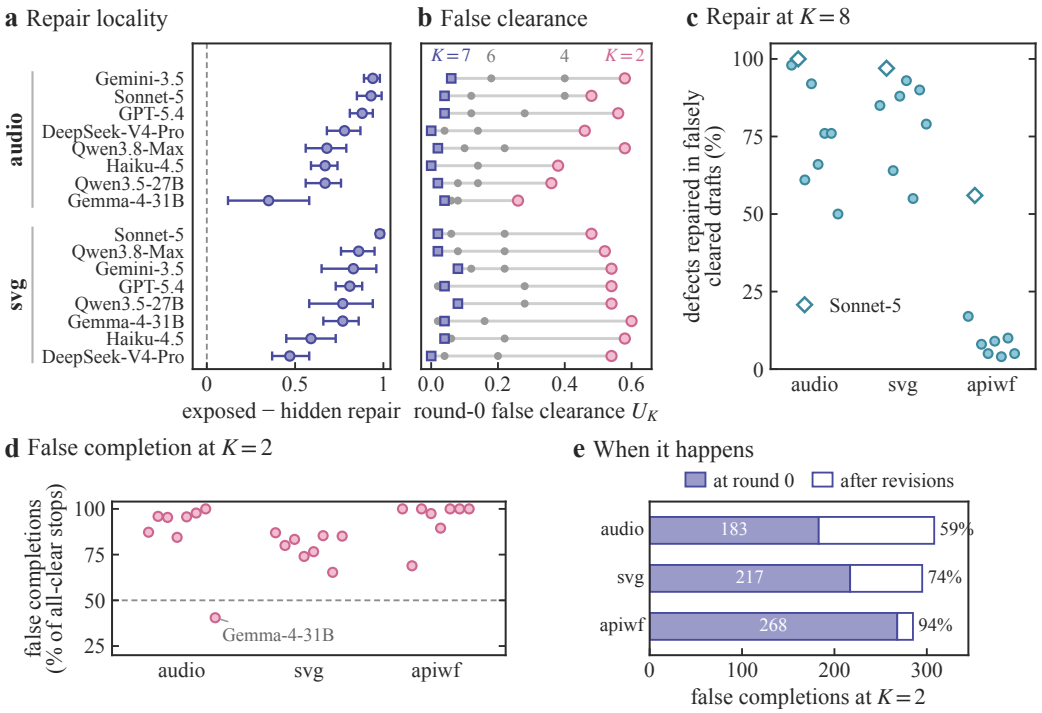

Figure 2: **Partial verification creates repair locality and false clearance.** (a) Repair locality (P1): the exposed-minus-hidden repair rate in the same trajectories, averaged over $K \in \{2, 4, 6, 7\}$; all 16 audio and svg model–domain cells are significant after Holm correction over all 24 cells (apiwf: Sec. 5.3). (b) Round-0 false clearance U_K for the same 16 cells; $U_2 > U_7$ in all 24 cells, including the 8 apiwf cells not plotted (Appendix Table 8). (c) Repair at K=8 of the defects in the drafts falsely cleared at K=2; one marker per model and domain. (d) Conditional false completion at K=2: the share of all-clear stops at which a hidden check still fails; one marker per model–domain cell (eight per domain), with the only cell below one half (dashed line) labeled. (e) The 888 false completions at K=2 by when they happen, at round 0 or after revisions; percentages: share at round 0. Bars: 95% task-bootstrap CIs.

5.3 Domain heterogeneity: API workflows resist repair

API workflows resist repair even when every check is reported. On the drafts falsely cleared at K=2, the K=8 runs start from the same drafts and report every check, yet they repair few of their defects in API workflows, against at least half in audio and SVG (Fig. 2c). Sonnet-5 is the one exception, repairing 56% in API workflows. The tasks are not unsolvable: every API-workflow task has a reference program that passes all eight checks (Appendix B). The split follows what a failing check tells the model. A failing audio or SVG check names one property of the artifact and usually gives its measured value and the required bound. The main API-workflow check reports which outcome is wrong but not which step of a long program caused it.

The obstacle is one check, `completeness`, which reports what went wrong but not where. It accounts for 80% of API-workflow round-0 defects and names the failed outcome assertion, such as a wrong final answer. Even when it is exposed, its defects are repaired in only 5% of trajectories, whereas checks that point to a specific API request are repaired in 42–100% (Appendix Table 13). Whether `completeness` is exposed therefore decides how an API-workflow loop ends. When it is hidden, nearly every run ends in an all-clear stop and almost all of these stops are false completions; when it is exposed, the loop rarely reaches an all-clear stop. This also explains why repair locality is weaker in API workflows: it is positive only when `completeness` is hidden (Appendix Table 14). For check design, this suggests pairing such an outcome check with checks that point to the failing step.

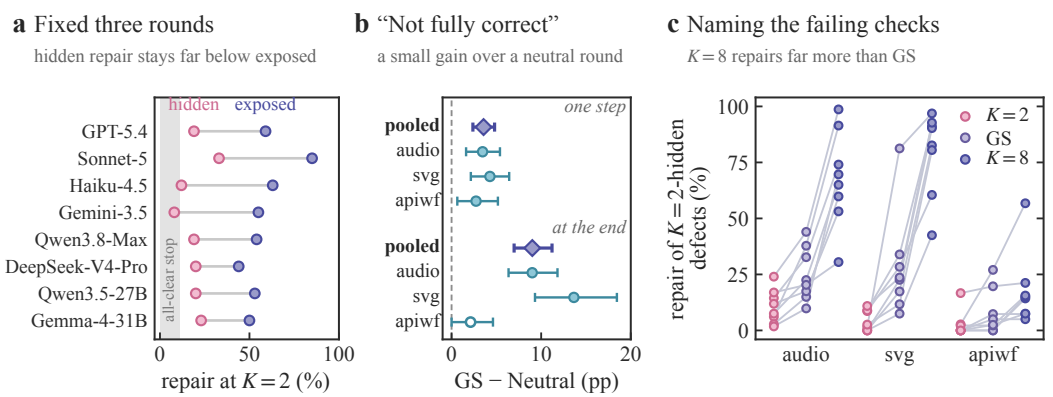

Figure 3: **Why the loop stops with defects left.** (a) With exactly three rounds and no all-clear stop, hidden defects at $K{=}2$ are still repaired far less often than exposed ones; the gray band is hidden repair under the all-clear stop (Appendix Table 7). (b) Matched continuation control: the "not fully correct" message (GS) minus a neutral offer of another round (Neutral), one step later and at the end, pooled over the three domains and by domain (domains Holm-corrected within each outcome; open markers: not significant). (c) Repair of the round-0 defects hidden at $K{=}2$, under $K{=}2$, the global-stop (GS) policy, and $K{=}8$; one line per model (Appendix Table 16). Bars: 95% CIs.

Coverage and cost. Higher coverage can improve efficiency: a $K{=}8$ trajectory costs 2.6× the tokens of $K{=}2$ but raises the fully correct rate from 12% to 48%, so each fully correct artifact costs 35% fewer tokens overall, and fewer for seven of eight models individually (Appendix E).

Robustness. The RQ1 results hold under four robustness analyses: (i) two fresh mask draws, (ii) a report without the instruction to fix the reported failures, (iii) the $K{=}2$ report padded to $K{=}8$ length, and (iv) two reruns with fresh round-0 drafts (Appendix B).

6 RQ2: Why does false completion occur?

Our experiments point to missing information rather than missing rounds: extra rounds recover few hidden defects (Sec. 6.1), a message that the artifact is not fully correct recovers a few more (Sec. 6.2), and naming the failing checks recovers far more (Sec. 6.3).

6.1 A three-domain fixed-horizon control

Stopping too early is not the main reason hidden defects remain. The three-domain fixed-horizon control runs every trajectory for exactly three revision rounds with no all-clear stop, at $K \in \{0, 2, 7\}$ on 10 tasks per domain, from the main grid's round-0 drafts and masks (Appendix C). Even without early stopping, hidden defects at $K{=}2$ are repaired far less often than exposed ones for every model (Fig. 3a). The rounds that early stopping skips add little: they raise the share of hidden round-0 defects repaired by at most 16 percentage points (Appendix Table 15). With equal rounds, coverage still matters: $K{=}7$ repairs more round-0 defects per task than $K{=}2$, net of checks that newly fail, for all eight models and significantly for five. With no report at all ($K{=}0$), three rounds of the bare instruction repair only a small share of round-0 defects, consistent with the limited self-correction without external feedback reported by Huang et al. (2024).

6.2 A "not fully correct" message helps, but only a little

Telling the model only that its artifact is not fully correct helps it repair hidden defects, but only a little. The matched continuation control tests this at *false-clear states*, where the exposed $K{=}2$ checks pass while a hidden check fails and the normal loop would stop. From each such state, two branches continue with the same report and revision directive and differ in one appended sentence, which each branch repeats at every later false-clear state. The *GS* branch gets *"The artifact is not*

fully correct yet. Please continue revising.", and the *Neutral* branch gets *"Another revision round is now available. Please continue revising."* Both sentences offer another round, and only the GS sentence says the artifact is still wrong, so the difference in repair measures the value of that information alone. The states come from the global-stop (GS) policy, which keeps the $K{=}2$ report, stops only when all eight checks pass or at the horizon, and appends the "not fully correct" sentence whenever the exposed checks pass but a hidden check fails. We ran this policy on every main-grid task as a third condition next to $K{=}2$ and $K{=}8$, and forked the two branches at each of the 904 first false-clear states it reached (Appendix C). The primary outcome is the repair, one revision later, of the hidden defects that were failing at the fork; repair at the end of the run is secondary.

The gain appears after a single revision and grows by the end of the run (Fig. 3b). After one revision, the branch told that its artifact is not fully correct repairs 9% of the hidden defects that were failing at the fork, against 6% in the Neutral branch, and the difference holds in each domain. By the end of the run, the GS branch repairs 18% against 9%, significantly more in audio and SVG but not in API workflows. The message therefore carries information beyond permission to continue. It does not, however, lower the residual defect count after one revision, because the extra repairs are offset by checks that passed at the fork and fail after the revision. By the end of the run, 81 GS branches and 28 Neutral branches are fully correct (Appendix C).

6.3 Naming the failing checks helps far more

On the same round-0 drafts, we compare $K{=}2$, the GS policy, and $K{=}8$ by what share of the $K{=}2$-hidden round-0 defects each repairs by the end of the run (Fig. 3c; Appendix Table 16). We compare repair rather than false completion, because the GS policy never stops while a hidden check fails, so its false completion is zero by design. $K{=}8$, which has the same stopping rule as GS but names every failing check, repairs more of these defects than GS in 20 of the 24 cells. GS repairs more than $K{=}2$ in 15 cells, and $K{=}8$ still beats GS in all of them, so in no cell does GS both beat $K{=}2$ and match $K{=}8$. GS shows no detectable harm: its fully correct rate is at or above that of $K{=}2$ in all 24 cells, and its repair of exposed defects shows no detectable change.

7 RQ3: Can models compensate for incomplete verification?

7.1 Reasoning helps drafting and repair, but coverage still matters

Reasoning roughly halves the effect of coverage but does not remove it. We ran each model with its built-in reasoning on and off on 15 tasks per domain, with the same tasks and masks in both arms (Appendix D). With reasoning on, residual defects still fall from $K{=}2$ to $K{=}8$ for every model, but overall the decline is half as steep (slope -0.43 vs. -0.84; Fig. 4a). Part of this comes from the drafts: with reasoning on, the round-0 draft has fewer defects (1.1 vs. 1.7), so less is left for coverage to fix. Measured per draft defect, coverage matters at least as much with reasoning as without under the loop's own stopping rule (Appendix Table 21). The slope shrinks in audio and SVG; in API workflows, it is shallow without reasoning but steeper with it, and reasoning detectably lowers residual defects only for Qwen3.8-Max and DeepSeek-V4-Pro (Appendix D). Reasoning lowers false completion at $K{=}2$ for every model but removes it for none.

Reasoning also helps the repair itself, most at low coverage. In a fixed-draft study, two branches revise the same reasoning-off draft for three rounds at $K{=}2$ and $K{=}8$, one with reasoning on in every revision call and one with it off (Appendix D). On the same draft, reasoning in the repair stage leaves fewer residual defects in every domain and for every model, and it narrows the gap between $K{=}2$ and $K{=}8$ from 0.56 to 0.31 defects per task (Fig. 4b). The narrowing holds in audio and SVG but not in API workflows (Appendix Table 23). Most of the extra repairs are of hidden defects, and newly introduced defects show no detectable increase overall (Appendix D).

7.2 Self-selected checks: targeting changes, repair does not

Letting the model choose which checks it sees does not detectably reduce residual defects compared with a random choice of the same number of checks. In each of 720 pairs, the model revises the same round-0 draft for two rounds, and each round's report shows two checks that no earlier report showed (Appendix D). In the self-selected arm the model picks the two checks in a separate call; in

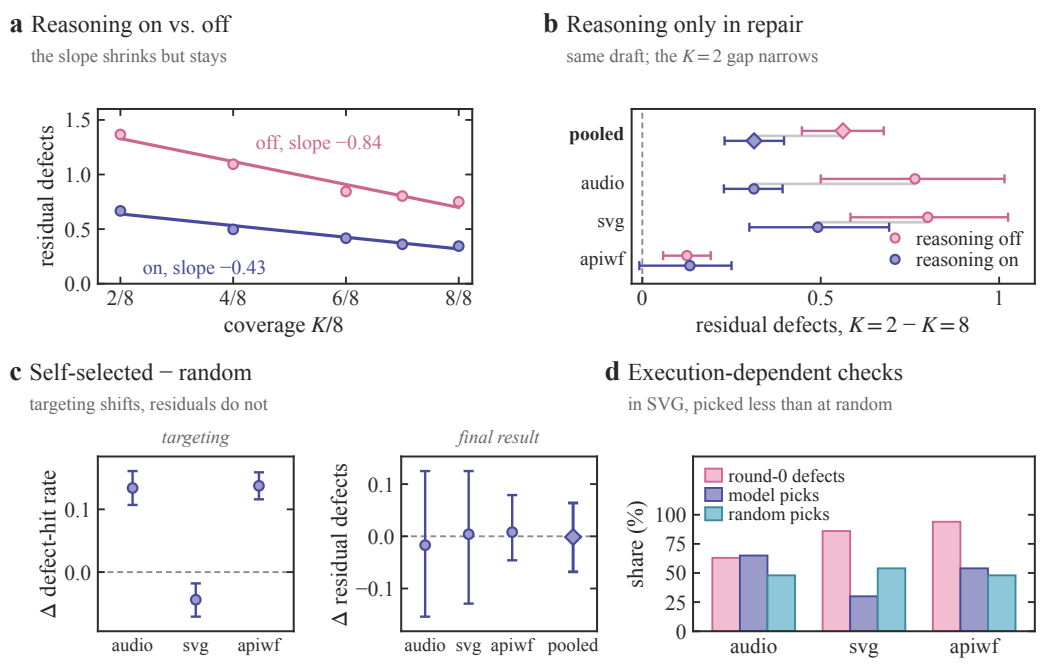

Figure 4: **Compensation is limited.** (a) Reasoning shrinks the slope of the residual defect count on $K/8$ without removing it (Sec. 7.1). (b) Revising the same reasoning-off draft with reasoning on narrows the residual gap between $K{=}2$ and $K{=}8$, pooled and in audio and SVG, but not in apiwf (Appendix Table 23). (c) Self-selected checks hit failing checks more often than random in audio and apiwf, less in SVG (left), without detectably fewer residual defects, pooled (diamond) or in any domain (right). (d) Share of checks that require rendering or running the artifact among round-0 defects, the models' picks, and random picks (Appendix Table 24). Bars: 95% CIs.

the random arm a fixed random schedule picks them, so both arms see 4 of the 8 checks over the same rounds. Residual defects show no detectable difference between the arms, pooled or in any domain (Fig. 4c). The conclusion holds for pairs whose round-0 draft had defects (Appendix D).

Self-selection changes which checks are exposed, not how many defects get repaired. The share of picked checks that were failing when picked is higher than under the random schedule in audio and API workflows and lower in SVG, all three detectably (Fig. 4c). The repair rate of the draft's defects, however, stays within about three percentage points of the random arm in every domain, and the numbers of fully correct artifacts are close (Appendix D).

In SVG, models pick the checks where most defects lie less often than a random choice does; these are the checks that require rendering or running the artifact. Such checks hold 86% of SVG round-0 defects but receive only 30% of the models' picks, against 54% under the random schedule (Fig. 4d). In API workflows, models pick such checks a little more often than at random, yet still far less often than their share of defects (Fig. 4d).

8 Conclusion

When the model cannot run the checks, partial verification concentrates repair on reported failures and lets the loop stop with hidden defects, often before any revision. Telling the model its artifact is not fully correct improves hidden-defect repair beyond a neutral offer of another round, but less than naming the failing checks. Reasoning improves drafts and repair without eliminating the effect of coverage. Model-chosen checks do not detectably reduce residual defects relative to the same number of random checks. Higher coverage lowers the token cost per fully correct artifact. What you don't check, models disproportionately fail to fix: a caution against trusting partial verification.

Reproducibility statement

Appendix A gives operational definitions of all 24 checks (eight per domain), the task construction of each domain, the exact prompt and feedback templates shown to the models (the variants of the follow-up studies are given with each study in Appendices B–D), and the model identifiers and decoding settings. The statistical procedures are stated in Sec. 4 and, per analysis, in Appendices B (RQ1), C (RQ2), and D (RQ3). We will release the evaluation harness, the task sets, the masks (the exposed checks for each task and K), the complete trajectory records of every experiment, and the analysis code.

Ethics statement

This study evaluates eight language models spanning proprietary and open-weight systems on artifact generation and repair tasks built from public data; no human subjects, personal data, or deployment decisions are involved. We report model failures under partial verification to inform safer evaluation practice.

AI use statement

Generative AI coding assistants were used, under author direction, mainly for the engineering of this study: implementing the evaluation harness, the experiment runners, and the analysis scripts, including data cleaning and reformatting; every reported number was recomputed from the stored experimental records. They were also used for finding related work, for drafting and editing text, and for figure-plotting code. They were not used to formulate or prove mathematical claims or to generate synthetic data, and the study involves no qualitative analysis; the research questions, experimental design, and interpretation of results are the authors' own. The eight evaluated models are the experimental subjects of the study. The authors reviewed all AI-assisted content and take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

References

Laksh Advani. From confident closing to silent failure: Characterizing false success in LLM agents. *arXiv preprint arXiv:2606.09863*, 2026.

Toufique Ahmed, Jatin Ganhotra, Avraham Shinnar, and Martin Hirzel. Investigating test overfitting on SWE-bench. In *Companion Proceedings of the 34th ACM International Conference on the Foundations of Software Engineering (FSE Companion)*, pp. 1292–1296, 2026. doi: 10.1145/3803437.3805574.

Anthropic. System card: Claude Haiku 4.5. https://www.anthropic.com/claude-haiku-4-5-system-card, October 2025. Accessed 2026-09-25.

Anthropic. System card: Claude Sonnet 5. https://www.anthropic.com/claude-sonnet-5-system-card, June 2026. Accessed 2026-09-25.

Johin Johny Arimbur. How many tries does it take? Iterative self-repair in LLM code generation across model scales and benchmarks. *arXiv preprint arXiv:2604.10508*, 2026.

Sidhesh Badrinarayan and Adithya Parthasarathy. Counterexamples as feedback for agent self-correction. *arXiv preprint arXiv:2609.02892*, 2026.

Mingguang Chen, Licheng Wang, and Bo Qu. Recursive self-improvement in AI: From bounded self-refinement to autonomous research loops. *arXiv preprint arXiv:2607.07663*, 2026a.

Siqi Chen, Xinyu Dong, Haolei Xu, et al. SVGenius: Benchmarking LLMs in SVG understanding, editing and generation. In *Proceedings of the 33rd ACM International Conference on Multimedia (ACM MM)*, 2025.

Weiliang Chen, Haowen Sun, Jun Gao, Jiawei Chi, Hanyang Wang, et al. HarnessEval-W: Agentifying the evaluation of visual worlds. *arXiv preprint arXiv:2608.16859*, 2026b.

Yizhe Chi, Wenyi Li, Deyao Hong, Xiaoqiu Wang, Mingju Gao, Kaisen Yang, Bingxiang He, Youjie Zheng, Calvin Xiao, and Qinhuai Na. AI4AI-Bench: Benchmarking LLM agents in algorithmic design for recursive self-improvement. *arXiv preprint arXiv:2608.20318*, 2026.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. URL https://arxiv.org/abs/2606.19348.

Youwang Deng. Goal-Autopilot: A verifiable anti-fabrication firewall for unattended long-horizon agents. *arXiv preprint arXiv:2606.11688*, 2026.

Sajib Acharjee Dip, Dawei Zhou, and Liqing Zhang. Think again or think longer? Selective verification for budget-aware reasoning. *arXiv preprint arXiv:2606.19808*, 2026.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Taco Cohen, and Gabriel Synnaeve. RLEF: Grounding code LLMs in execution feedback with reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 19034–19055, 2025.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026. URL https://arxiv.org/abs/2607.02770.

Francesca Gomez. Can escalation channels redirect reward hacking toward defect disclosure? *arXiv preprint arXiv:2608.29460*, 2026.

Google DeepMind. Gemini 3.5 Flash - model card. https://deepmind.google/models/model-cards/gemini-3-5-flash/, May 2026. Accessed 2026-09-25.

Xin He, Yanlin Wang, Mingwei Liu, Jiachi Chen, Hongyu Zhang, and Guanbin Li. SWE-Gate: Passing functional tests is not enough for software engineering agents. *arXiv preprint arXiv:2609.04167*, 2026.

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 *International Conference on Learning Representations (ICLR)*, 2024.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Divake Kumar, Sina Tayebati, Devashri Naik, Amanda Sofie Rios, Nilesh Ahuja, Omesh Tickoo, Ranganath Krishnan, and Amit Ranjan Trivedi. CURA: Certified runtime alarms for computer-use agents. *arXiv preprint arXiv:2608.27808*, 2026.

Bojie Li and Noah Shi. Interaction scaling: Grounding the third axis of test-time compute. *arXiv preprint arXiv:2607.11598*, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Phillip Long, Zachary Novack, Taylor Berg-Kirkpatrick, and Julian McAuley. PDMX: A large-scale public domain MusicXML dataset for symbolic music processing. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

OpenAI. GPT-5.4 Thinking system card. https://openai.com/index/gpt-5-4-thinking-system-card/, March 2026. Accessed 2026-09-25.

Dipkumar Patel. Anytime verified agents: Adaptive compute allocation for reliable LLM reasoning under budget constraints. *Transactions on Machine Learning Research (TMLR)*, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, February 2026a. Accessed 2026-09-25.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork. https://qwen.ai/blog?id=qwen3.8, August 2026b. Accessed 2026-09-25.

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

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Edward K. Smith, Earl T. Barr, Claire Le Goues, and Yuriy Brun. Is the cure worse than the disease? Overfitting in automated program repair. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, 2015.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations of large language models on reasoning and planning tasks. In *International Conference on Learning Representations (ICLR)*, 2025.

Yueh Tang and Justin Zhan. Verify what matters: Budgeted verification for tool-using agents under counterfactual downstream harm. *Transactions on Machine Learning Research (TMLR)*, 2026. URL https://openreview.net/forum?id=nv1jzr0FaZ.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Yuvraj Verma. Try again, don't look back: Blind resampling outperforms self-repair in small code models. *arXiv preprint arXiv:2607.26117*, 2026.

Jiaming Wang, Yunke Zhao, Peng Ding, Jun Kuang, Yibin Shen, Zhe Tang, Yilin Jin, ZongYu Wang, Xiaoyu Li, and Xuezhi Cao. Meeseeks: A feedback-driven, iterative self-correction benchmark evaluating LLMs' instruction following capability. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14745–14773, 2026. doi: 10.18653/v1/2026.findings-acl.725.

Xiaonan Xu and Wenjing Wu. Validation evidence in LLM repair agents: How much of what passes actually tests the bug? *arXiv preprint arXiv:2607.28871*, 2026.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. AgenTracer: Who is inducing failure in the LLM agentic systems? In *International Conference on Learning Representations (ICLR)*, 2026a.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel Machine: Open-ended evolution of self-improving agents. In *International Conference on Learning Representations (ICLR)*, 2026b.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *International Conference on Machine Learning (ICML)*, 2024.

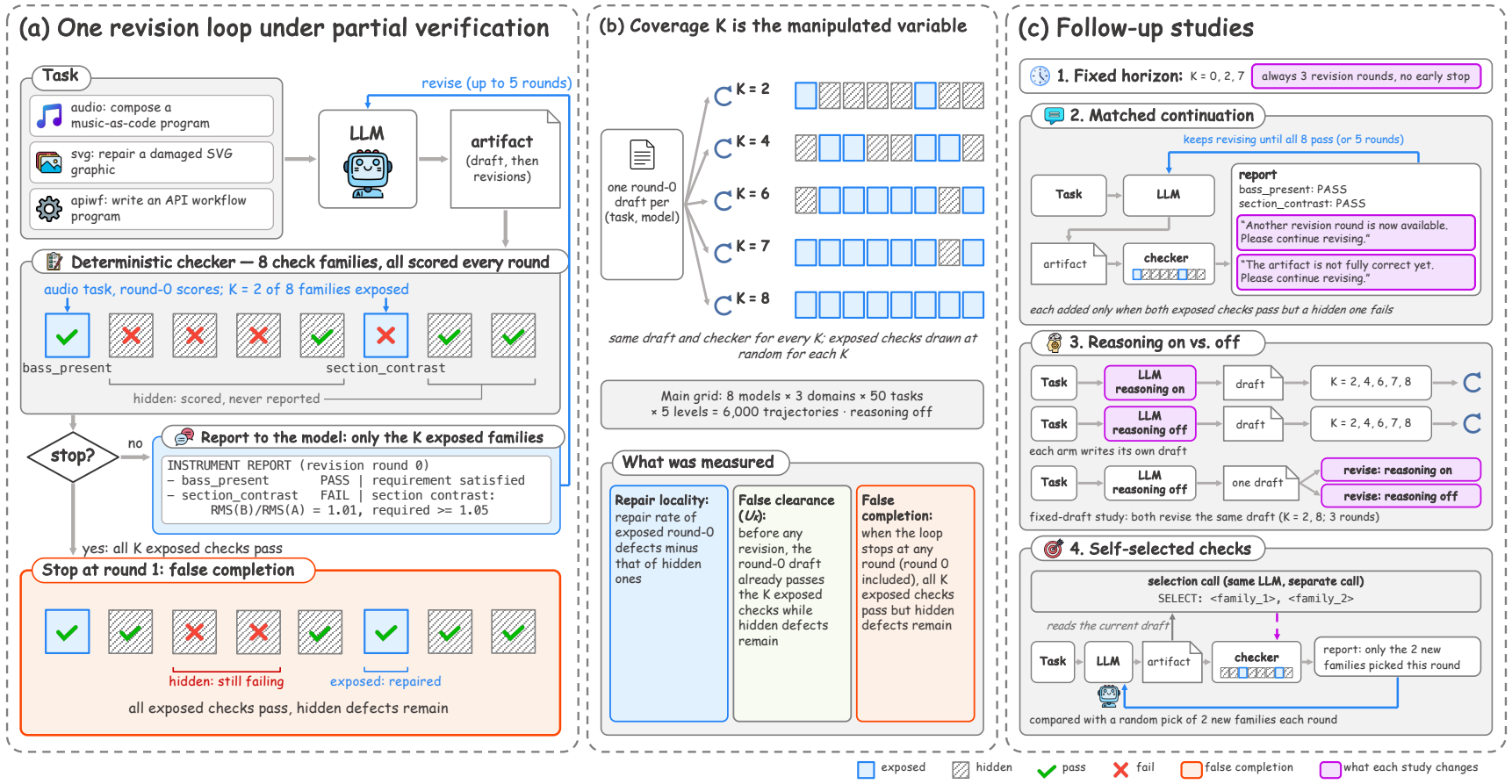

Figure 5: Full-text version of Figure 1: the task descriptions, the round-0 report as the model receives it (the audio example of Appendix F), the main grid, and each follow-up study drawn with all of its labels; like the prompts and reports, this figure calls each check a *family*.

A HARNESS DETAILS

Checks. Every task carries eight deterministic checks (Table 1; the prompts, report templates, and Figure 5 call each check a *family*); each check is a deterministic program returning pass or fail and, on failure, a structured counterexample. Full operational definitions: Tables 3–5.

Domain	Checks (8 per task)
audio	bass_present, density, melody_range, no_clipping, scale_conformance, section_contrast, stepwise, tempo
svg	accessibility, attributes, containment, contrast, coverage, ids_preserved, structure, visibility
apiwf	argument_fidelity, authentication_first, completeness, dependency_order, entity_binding, exactly_once, precondition_reads, side_effect_conservation

Table 1: Checks per domain.

	audio	**svg**	**apiwf**
Artifact	music-as-code program	SVG document	API-call workflow program
Ground truth	PDMX musical scores	original graphic (SVGenius)	official sandbox execution (AppWorld)
Model's job	compose from the specification	repair the damaged graphic	write the program from scratch
Failing check	names one property, usually with its measured value and the required bound	names one property, usually with its measured value and the required bound	the main check, completeness, names the wrong outcome, not the step that caused it

Table 2: Domain overview; all checks are deterministic and all eight checks are scored every round.

Feedback template and protocol. Each exposed check contributes exactly one report line of the same format, `- <family> <STATUS> | <witness>: <STATUS>` is `PASS` or `FAIL`, and the witness is the check's structured counterexample on `FAIL` and "requirement satisfied" on `PASS`, padded with trailing spaces to a fixed width so that a check's passing line is as long as its failing line. There is at most one counterexample per failing check per round and nothing (no verdict, measurement, or count) for hidden checks. The revision directive is fixed (quoted under "Prompt and feedback templates" below).

Operational check definitions. Tables 3–5 give the operational pass condition for each of the 24 checks. Six audio checks use per-task targets or bounds derived from each task's source score, and `no_clipping` and `bass_present` use the same thresholds in every task; the four render-based SVG checks (`containment`, `visibility`, `contrast`, `coverage`) operate on a rasterized rendering (fixed canvas; anti-aliased boundaries handled with an alpha mask threshold of 128 and a small fixed fraction tolerance on area ratios), and the other four inspect the SVG markup directly; the first six API-workflow checks of Table 5 are evaluated from the program's request trace and captured execution output, and `completeness` and `side_effect_conservation` from the benchmark's own labeled assertion suite and write-scope guard. Checks without a numeric threshold are exact/Boolean. In every domain a hard validity failure (unparseable artifact; malformed XML or missing `viewBox`; program rejected by our static code scan for file access, system modules such as `os` or `subprocess`, `eval`/`exec`, or reflection) counts against all eight checks. P1 and the within-check comparison (Appendix B) instead use only trajectories whose round-0 draft and terminal artifact are both parseable, because for an unparseable draft a check that passes later reflects a repaired output format rather than a repaired property; this leaves out 130 of the 1,600 audio trajectories at $K \in \{2, 4, 6, 7\}$, 8 of the 1,600 apiwf trajectories, and no svg trajectory. The `no_clipping` check fails once the raw mix's peak amplitude reaches 0.999, but its fixed failure message, which the model receives verbatim whenever the check is exposed and fails, reads `clipping: peak amplitude >= 1.0 in raw mix; reduce`

`velocities or track count` and gives no measured value, so for peaks in $[0.999, 1.0)$ its stated bound is inaccurate. In the main grid this applies to 7 of the 2,847 scored `no_clipping` failures that carry this message (0.25%; hard validity failures excluded; each shared round-0 draft counted once) and to 6 of the 2,887 such reports shown to a model (0.21%); the verdicts are unaffected, and all analyses use the feedback exactly as the models received it.

Check (audio)	Property	Pass condition
tempo	rendered tempo	audio-estimated tempo within a per-task tolerance of the target (e.g., $\pm 4\%$)
no_clipping	waveform headroom	peak amplitude of the raw mix < 0.999 (exact)
bass_present	low-frequency content	share of the magnitude spectrum below 200 Hz strictly > 0.0246 (fixed calibrated threshold; the prompt calls it an energy ratio)
scale_conformance	tonality	out-of-scale note count $\leq$ per-task maximum
melody_range	melodic span	melody-track pitch span (semitones) $\leq$ per-task maximum
section_contrast	dynamics	RMS(section B)/RMS(section A) $\geq$ per-task minimum ratio
stepwise	melodic motion	fraction of melody intervals $\leq$ 2 semitones $\geq$ per-task minimum
density	rhythmic density	total note count across all tracks, divided by the task's specified beat count (bars $\times$ 4), $\geq$ per-task minimum

Table 3: Audio checks. On failure, the counterexample reports the measured value and the violated bound, except for `no_clipping`, whose fixed message gives no measured value and states the bound as 1.0 (see the operational definitions above).

Check (svg)	Property	Pass condition
structure	grouping	named group contains exactly the required member elements (exact)
attributes	attribute values	named element's attribute equals the required value (exact)
ids_preserved	identity stability	required id set present after editing (exact)
accessibility	metadata	required accessibility elements (title/description) present (exact)
containment	layout	element's rendered mask lies fully inside the viewport minus a per-task pixel margin
visibility	occlusion	visible composited area / isolated in-viewport area $\geq$ per-task minimum fraction
contrast	legibility	relative-luminance contrast ratio between two elements' median composited colors $\geq$ per-task minimum
coverage	area	element's rendered area fraction of the canvas within a per-task [lo, hi] band

Table 4: SVG checks. Render-based checks use isolated and composited raster masks; the counterexample reports the measured quantity.

Check (apiwf)	Property	Pass condition
authentication_first	auth ordering	for every app used, an auth request precedes that app's other requests (trace-exact)
precondition_reads	read-before-write	each app receives a read before its first write (trace-exact)
entity_binding	write-target and identity binding	every non-auth write request targets an app within the task's declared write scope (declared apps plus the bookkeeping apps `supervisor`, `admin`, and `api_docs`), and every authentication uses the identity of the task's supervisor, the AppWorld user on whose behalf the program acts (trace-exact)
argument_fidelity	argument validity	no requests failing with argument-class errors (fixed error classification)
dependency_order	step ordering	no state-class errors indicating an operation before its prerequisite
exactly_once	idempotence	no two non-authentication write requests have the same method, URL, and payload (access token ignored) (exact)
completeness	task outcome	the benchmark's labeled assertion suite for the task passes (exact)
side_effect_conservation	write scope	no state changes outside the apps declared by the task's assertions (exact)

Table 5: API-workflow checks. Checks 1–6 are process properties over the executed request trace; 7–8 are outcome properties over the sandbox world state.

Task construction. **Audio**: each of the 50 tasks is derived from a real public score (PDMX). Six of the eight audio checks use targets or bounds derived from the source score's measured attributes; `no_clipping` and `bass_present` use the same thresholds in every task. The `section_contrast` bound is $\max(1.05, 0.9r)$, where r is the source score's own RMS(B)/RMS(A) ratio; the 1.05 floor applies in 41 of the 50 tasks. The model writes a program in a small audio DSL from the specification; the round-0 draft (and hence its defect state) is the model's own first attempt, cached and shared across all conditions. The source score itself, converted to this DSL, passes all eight checks in every task. **SVG**: each task pairs a pristine source graphic (drawn from SVGenius source pools, stratified by source difficulty) with a starting artifact derived from it in which a recorded set of checks is violated (four to six per task; see the next paragraph); requirements (element ids, attribute targets, thresholds) are task parameters written into the specification text, and the model edits the starting artifact. The pristine graphic passes all eight checks in every task. **API workflows**: the 50 tasks are new instances of ten AppWorld task families spread over five apps (tasks per app: venmo 22, spotify 15, phone 6, simple_note 4, splitwise 3); each keeps the family's world and instruction wording and changes one instruction parameter (for example the time window, library type, or contact group), and no instance repeats an official AppWorld task. An instance is admitted only if, in a fresh sandbox, a reference program passes both the family's own AppWorld evaluation script and all eight of our checks; that program is the task's reference solution. The model writes a complete program against a fixed, checksummed API manual, and the round-0 draft is its own first attempt. In all domains the task set and specification text are held fixed across K conditions, and each (task, K) mask is generated in advance and held fixed.

SVG checks that pass from the start. Each SVG task breaks four to six of the eight checks in the starting graphic, one edit per check (the *edited* checks), and leaves the other two to four *intact*, as in real repair, where an artifact usually violates some requirements and already meets others. The specification lists all eight requirements, so a draft can also break an intact check (7% of intact checks fail in round-0 drafts). This design bears on the false clearance U_2 reported in Sec. 5.2. Because each task has one $K{=}2$ mask for all models, a mask that exposes intact checks makes a round-0 all-clear stop easier: pooled over the eight models, U_2 is 70% (67 of 96 drafts) when both exposed checks are intact, 54% (107 of 200) when one is, and 41% (43 of 104) when both are edited, against 4% (16 of 400) at $K{=}7$. The hidden defects in these 217 falsely cleared drafts are real failing checks: 328 edited checks the draft left unrepaired and 37 intact checks the draft broke. The $K{=}2$ loop stops at round 0 and repairs none of them. Measured at the terminal state (terminal repair, Sec. 6.2), continuing from the same drafts without naming the failing checks repairs 16% of them in the Neutral branch and 26% in the GS branch of the matched continuation control, and revising the same drafts at $K{=}8$, where every check is reported, repairs 80%. These rates do not differ detectably between drafts whose exposed checks were intact and drafts whose exposed checks were

edited (the 95% task-bootstrap interval of the difference includes zero in each case), and continuing breaks fewer than 2% of the intact checks that passed at round 0.

Prompt and feedback templates. System prompt (all domains): *"You are an expert engineer. Satisfy every requirement in the task exactly. When you receive an instrument report, revise your artifact to fix the reported failures."* Initial prompts: for audio, *"Write a program in the audio DSL that satisfies EVERY requirement below."* followed by the DSL description and the task specification, which gives every numeric threshold except the clipping bound (for clipping it says only *"The audio must not clip."*); for svg, *"Below is an SVG artifact that violates some of the requirements. Edit it so that EVERY requirement is satisfied. Keep the artifact's overall design; make targeted fixes."* followed by requirements and the starting artifact; for apiwf, *"You are operating a personal digital assistant for a supervisor."* followed by the task instruction, current date-time, execution rules (login per app, use only manual APIs, call `complete_task()` when done), and the full API manual. Feedback each round is the check report of Sec. 4, headed *"INSTRUMENT REPORT (revision round r)"*, where r is the round of the artifact being reported on (0 for the round-0 draft), and followed by one line per *exposed* check (`- <family> <STATUS> | <witness>`) and the revision directive *"Revise the artifact to fix the reported failures. Return the complete revised artifact in a single fenced code block."* The global-stop policy (Sec. 6.2) appends the "not fully correct" message whenever the exposed checks pass but a hidden check fails, exactly: *"The artifact is not fully correct yet. Please continue revising."* In the matched continuation control, the GS branch uses this message and the Neutral branch puts *"Another revision round is now available. Please continue revising."* in its place. In the two fixed-horizon controls (Appendices B and C), a round at $K{=}0$, or a round in which every exposed check passes, gets no report, only the bare revision instruction *"Revise the artifact. Return the complete revised artifact in a single fenced code block."* The self-selection study issues a separate, stateless selection call per round (context: task, current artifact, the checks already exposed and their results, and the list of not-yet-exposed checks), whose response is constrained to one line, *"SELECT: ⟨family$_1$⟩, ⟨family$_2$⟩"*, and never enters the revision conversation. The selection call runs with reasoning disabled, and this single line is its whole visible answer; selection policies that deliberate explicitly, or that run under a model's reasoning mode, are untested.

Models, APIs, and execution. Model-facing identifiers and routes: `gpt-5.4`, `claude-sonnet-5`, and `claude-haiku-4-5` via an institutional API proxy; `gemini-3.5-flash` via Google's API; `qwen3.8-max` via Alibaba Bailian; `deepseek-v4-pro` via DeepSeek's API; `Qwen3.5-27B` and `gemma-4-31B-it` (open weights) served in bf16 either from our own vLLM deployment (Hugging Face revisions `fc05daec` and `842da379`) or through provider APIs (Bailian for Qwen3.5-27B; a bf16-locked Crusoe route and Novita's bf16 endpoint, directly or through OpenRouter, for Gemma-4-31B). The revision calls of a trajectory always come from one *environment* (serving backend: our own deployment or one provider route). Gemma-4-31B's main grid ran on our own vLLM deployment, and three of its follow-up studies (the reasoning two-arm study and the fixed-draft repair study of Appendix D, and the three-domain fixed-horizon control of Appendix C) ran their revision calls on the Crusoe route. The two reasoning studies also drew their round-0 drafts there (the fixed-draft repair study reuses the two-arm study's reasoning-off drafts), so neither pairs arms from different environments; the three-domain fixed-horizon control revises the main-grid drafts, so Gemma-4-31B's stopping component (Table 15, 22 pairs) compares Crusoe revisions with main-grid revisions from our deployment. Qwen3.5-27B's main-grid drafts all come from our own deployment and are shared by a task's five K conditions; within each task, the revision calls of all K conditions use one environment (Bailian for 64 tasks, our deployment for 74; the other 12 of its 150 tasks stopped at round 0 under every K and made no revision call), so 264 of the 535 trajectories that made revision calls pair a self-hosted draft with Bailian revisions. Its P1 is $+0.67$, $+0.77$, and $+0.11$ in audio, svg, and apiwf; restricted to the 74 tasks served entirely by our deployment, it is $+0.73$ $[+0.59, +0.87]$, $+0.96$ $[+0.88, +1.00]$, and $+0.11$ $[+0.04, +0.18]$. Gemma-4-31B's revision calls in the global-stop policy, the matched continuation control, the self-selection study, and most robustness analyses of Appendix B came from Novita's bf16 endpoint (directly or through OpenRouter) rather than our deployment, and Qwen3.5-27B's global-stop runs came from Bailian, so these models' comparisons with the main grid in those analyses also cross serving backends. Unless stated otherwise, experiments run with reasoning disabled and `max_tokens` 16,384; the selection call of the self-selection study is capped at 32 tokens. The two reasoning studies of

Appendix D (the two-arm study and the fixed-draft repair study) set `max_tokens` to 32,768 in both arms, as does the reasoning-on arm of that appendix's replay diagnostic, with the built-in reasoning modes listed there. In the two reasoning studies no call of the other six models exceeded this cap (some Gemini-3.5 and DeepSeek-V4-Pro calls were cut off at it), but the reasoning of the two Qwen models did: in the reasoning-on arm, Qwen3.8-Max exceeded it on 121 of 312 call records in the two-arm study and on 50 of 270 revision calls in the fixed-draft study, and Qwen3.5-27B on 11 of 765 and 1 of 270. All runs use temperature 0, top-p 1, and seed 0 offered per call with acceptance as in Table 6: Sonnet-5 accepted none of the three and ran at the provider's defaults, GPT-5.4 accepted only the seed, Gemini-3.5 only temperature and top-p, and Haiku-4.5 only temperature. The rerun panel (Table 12) bounds the variance this leaves: over three runs of 15 tasks per model (the main-grid run and two reruns, each rerun with its own fresh round-0 drafts), Sonnet-5's terminal residual defect count moved by at most 2, and in 117 of the 120 (model, task) pairs the K=2-minus-K=8 difference in terminal residual defects was never positive in one run and negative in another. Apart from the weight revisions pinned on our own deployment, no immutable model-snapshot identifier is available (the endpoints return none); we therefore record the served-model string returned per call, which was single-valued within every trajectory and every paired arm. Recorded reasoning-token telemetry was zero across all 1,305 Sonnet-5 main-grid revision calls. All experiments were run in August–September 2026.

Model	temperature=0	top_p=1	seed=0
Qwen3.8-Max	accepted	accepted	accepted
GPT-5.4	—	—	accepted
Gemini-3.5	accepted	accepted	—
Haiku-4.5	accepted	—	—
Sonnet-5	—	—	—
DeepSeek-V4-Pro	accepted	accepted	not offered
Qwen3.5-27B	accepted	accepted	accepted
Gemma-4-31B	accepted	accepted	accepted

Table 6: Decode-parameter acceptance per model. Each parameter is offered on every call where the interface exposes it, with acceptance logged per request: "accepted" = applied; "—" = offered but not accepted (server default applies); "not offered" = no such parameter in the interface. The rerun panel (Table 12) bounds the run-to-run variance that server defaults leave.

B RQ1 SUPPLEMENTS AND ROBUSTNESS

This appendix supports Sec. 5. Tables 7 and 8 give the main-grid results per model–domain cell behind Secs. 5.1 and 5.2. The next three paragraphs are robustness analyses (i)–(iii) of Sec. 5 (mask redraw; directive removal, which removes the instruction to fix the reported failures; the length-matched report), followed by the fixed-horizon control for repair locality (the test in Sec. 5.1 of whether early stopping explains the gap) and the rerun panel of analysis (iv) (Table 12). The remaining paragraphs give the within-check comparison (the test in Sec. 5.1 of whether exposed checks are inherently easier to repair), further descriptives of the main grid, and the API-workflow analyses of Sec. 5.3, which end with the API-workflow P1 values and the Holm-corrected P1 tests for all 24 cells.

Domain	Model	P1 [95% CI]	exp. vs hid. @$K2$	U_K: $K2{\to}4{\to}6{\to}7$
audio	Gemini-3.5	$+0.94$ [$+0.89$, $+0.98$]	94% vs 2%	.58/.40/.18/.06
audio	Sonnet-5	$+0.93$ [$+0.85$, $+0.99$]	96% vs 4%	.48/.40/.12/.04
audio	GPT-5.4	$+0.88$ [$+0.81$, $+0.94$]	96% vs 1%	.56/.28/.12/.04
audio	DeepSeek-V4-Pro	$+0.78$ [$+0.68$, $+0.87$]	89% vs 2%	.46/.14/.04/.00
audio	Qwen3.8-Max	$+0.68$ [$+0.56$, $+0.79$]	86% vs 4%	.58/.22/.10/.02
audio	Haiku-4.5	$+0.67$ [$+0.59$, $+0.74$]	79% vs 3%	.38/.14/.00/.00
audio	Qwen3.5-27B	$+0.67$ [$+0.56$, $+0.76$]	72% vs 1%	.36/.14/.08/.02
audio	Gemma-4-31B	$+0.35$ [$+0.12$, $+0.58$]	40% vs 0%	.26/.08/.06/.04
svg	Sonnet-5	$+0.98$ [$+0.96$, $+1.00$]	100% vs 2%	.48/.22/.06/.02
svg	Qwen3.8-Max	$+0.86$ [$+0.76$, $+0.95$]	79% vs 0%	.52/.22/.08/.02
svg	Gemini-3.5	$+0.83$ [$+0.65$, $+0.96$]	100% vs 9%	.54/.22/.12/.08
svg	GPT-5.4	$+0.81$ [$+0.73$, $+0.88$]	78% vs 0%	.54/.28/.02/.04
svg	Qwen3.5-27B	$+0.77$ [$+0.58$, $+0.94$]	92% vs 11%	.54/.28/.08/.08
svg	Gemma-4-31B	$+0.77$ [$+0.66$, $+0.86$]	83% vs 3%	.60/.16/.02/.04
svg	Haiku-4.5	$+0.59$ [$+0.45$, $+0.73$]	60% vs 9%	.58/.22/.06/.04
svg	DeepSeek-V4-Pro	$+0.47$ [$+0.37$, $+0.58$]	57% vs 0%	.54/.20/.04/.00

Table 7: Main-grid results for the 16 audio and svg model–domain cells (95% task-bootstrap CIs, B=10k). P1 (repair locality, Sec. 3) is the exposed-minus-hidden repair-rate difference computed at each $K \in \{2, 4, 6, 7\}$ and averaged with defect-count weights; the "exp. vs hid." column gives the repair rates of exposed and of hidden round-0 defects at K=2. The last column lists U_K at K=2, 4, 6, 7; of this sequence we claim only $U_2 > U_7$ (tested in Table 8). API-workflow P1 values are listed under "API-workflow check granularity" below.

Model	audio $U_2 - U_7$	svg $U_2 - U_7$	apiwf $U_2 - U_7$	apiwf U_K: $K2{\to}4{\to}6{\to}7$
GPT-5.4	$+.52$ [.38,.66]	$+.50$ [.36,.64]	$+.52$ [.36,.66]	.58/.38/.16/.06
Sonnet-5	$+.44$ [.30,.58]	$+.46$ [.32,.60]	$+.44$ [.28,.60]	.54/.32/.18/.10
Haiku-4.5	$+.38$ [.24,.52]	$+.54$ [.38,.68]	$+.62$ [.48,.76]	.70/.46/.20/.08
Gemini-3.5	$+.52$ [.38,.66]	$+.46$ [.32,.60]	$+.66$ [.52,.80]	.76/.50/.24/.10
Qwen3.8-Max	$+.56$ [.42,.70]	$+.50$ [.36,.64]	$+.54$ [.38,.68]	.62/.40/.14/.08
DeepSeek-V4-Pro	$+.46$ [.32,.60]	$+.54$ [.40,.68]	$+.56$ [.40,.72]	.66/.40/.16/.10
Qwen3.5-27B	$+.34$ [.22,.48]	$+.46$ [.30,.60]	$+.64$ [.48,.78]	.74/.46/.22/.10
Gemma-4-31B	$+.22$ [.12,.34]	$+.56$ [.42,.70]	$+.64$ [.48,.78]	.76/.50/.26/.12

Table 8: False clearance U_K (P2, Sec. 3) is the share of tasks whose shared round-0 draft passes all K exposed checks while a hidden check fails. The audio, svg, and apiwf $U_2 - U_7$ columns give this contrast with 95% task-bootstrap CIs in all 24 model–domain cells; every lower bound is positive. The last column lists the apiwf U_K at K=2, 4, 6, 7 (audio and svg: Table 7).

Mask redraw. Each (task, K) has a single random mask, so the RQ1 results could depend on which checks that one draw happens to expose; robustness analysis (i) of Sec. 5 redraws the masks. Two fresh mask draws (seeds 4407, 5407; 3,200 new trajectories: 50 tasks $\times$ $K \in \{2, 7\}$ in each of the 16 audio/svg model–domain cells per draw) re-estimate three outcomes per model–domain cell; with the main-grid masks, each cell and outcome thus has three estimates, one per *mask set*. The first is R_2, a task-level exposed-minus-hidden repair difference at K=2: for each task whose round-0 draft has at least one exposed and one hidden defect, the repair rate of its exposed round-0 defects minus that of its hidden ones at the terminal state, averaged over those tasks with equal weight, with an unparseable artifact counted as failing every check. Unlike P1 it is neither defect-weighted nor averaged over K, so its main-grid values differ from Table 7. The other two are K=2-minus-K=7 differences: ΔF, in the share of all trajectories that end in false completion (all tasks are the denominator, not only all-clear stops), and ΔD, in mean terminal residual defects. All 144 point estimates (16 cells $\times$ 3 outcomes $\times$ 3 mask sets) share the main-grid sign. For each cell and outcome, the difference between every two of the three mask sets has a task-paired bootstrap CI (144 CIs); 140 of them include zero (the exceptions: one R_2 and one ΔD difference for DeepSeek-V4-Pro svg, and one ΔD difference each for DeepSeek-V4-Pro and Qwen3.5-27B audio; every ΔF CI includes zero); the largest between-mask difference anywhere is $+0.92$ [$+0.24$, $+1.68$] in ΔD (audio, DeepSeek-V4-Pro; main-grid masks minus seed 4407). Across the 16 cells, R_2 ranges from

$+0.23$ to $+0.94$ under the main-grid masks, from $+0.14$ to $+0.93$ under seed 4407, and from $+0.05$ to $+0.91$ under seed 5407. The redraws were not run in apiwf; there, the round-0 contrast $U_2 - U_7$ depends only on the shared drafts and the masks, so it is recomputed under both redrawn mask sets from the existing drafts: all 24 estimates (8 models $\times$ 3 mask sets) are positive ($+0.44$ to $+0.66$), and every between-mask paired CI includes zero (largest difference 0.10). `completeness`, the check that decides how apiwf trajectories end (Sec. 5.3), is exposed at $K{=}2$ in 12, 13, and 13 of the 50 apiwf tasks under the main-grid masks, seed 4407, and seed 5407.

Directive-removal control. Both the system prompt and the revision directive tell the model to fix the reported failures; robustness analysis (ii) of Sec. 5 asks whether this instruction, rather than the report itself, steers repair toward exposed checks. For every model, each task whose round-0 draft has at least one exposed and at least one hidden defect under the $K{=}2$ mask was revised from the same cached draft in two arms under the main grid's $K{=}2$ protocol (all-clear stop, five-round horizon): arm O keeps the original system prompt and directive; arm N removes the system prompt's last sentence ("When you receive an instrument report, revise your artifact to fix the reported failures.") and reduces the directive to "Revise the artifact. Return the complete revised artifact in a single fenced code block."; the check report is attached in both arms. Table 9 gives the counts. In audio and svg, hidden repair moves by -1.6 to $+7.0$ pp and exposed repair by -10 to $+3$ pp across the eight models (N minus O); no model's paired hidden-repair difference has a CI excluding zero (task bootstrap, $B{=}10{,}000$). In apiwf, with 43 eligible (model, task) units over 24 distinct tasks (Gemini-3.5 has none), hidden repair is 5/55 with the directive and 10/55 without it, $+0.091$ $[-0.045, +0.255]$ pooled over models (bootstrap over the 24 tasks, each carrying all of its model units); the net gain of five repairs comes from one DeepSeek-V4-Pro task (elsewhere, one more repair for Qwen3.8-Max and one fewer for Sonnet-5 cancel out), a task whose round-0 reply contains no program and therefore fails all eight checks (Appendix A), and exposed repair (16/45 vs. 17/45) and false completions (15/43 vs. 16/43, again with vs. without the directive) barely move.

Model	n	exposed repair		hidden repair		false compl.		hidden N−O [95% CI]
		O	N	O	N	O	N	
GPT-5.4	33	30/37	31/37	1/57	5/57	24/33	26/33	$+0.070$ [$+0.000$, $+0.200$]
Sonnet-5	41	44/45	45/45	17/69	17/69	39/41	39/41	$+0.000$ [-0.042, $+0.044$]
Haiku-4.5	45	41/50	37/50	11/87	12/87	31/45	29/45	$+0.011$ [$+0.000$, $+0.038$]
Gemini-3.5	24	25/31	24/31	19/57	20/57	20/24	19/24	$+0.018$ [-0.053, $+0.098$]
Qwen3.8-Max	34	34/40	32/40	15/61	14/61	27/34	26/34	-0.016 [-0.077, $+0.036$]
DeepSeek-V4-Pro	40	32/45	32/45	2/81	2/81	28/40	28/40	$+0.000$ [-0.037, $+0.036$]
Qwen3.5-27B	39	36/52	34/52	27/114	26/114	25/39	23/39	-0.009 [-0.028, $+0.000$]
Gemma-4-31B	23	20/30	17/30	11/61	14/61	16/23	12/23	$+0.049$ [$+0.000$, $+0.101$]
API workflows								
GPT-5.4	15	4/15	4/15	0/15	0/15	4/15	4/15	$+0.000$ [$+0.000$, $+0.000$]
Sonnet-5	4	4/4	4/4	2/5	1/5	3/4	4/4	-0.200 [-0.750, $+0.000$]
Haiku-4.5	3	3/3	3/3	0/3	0/3	3/3	3/3	$+0.000$ [$+0.000$, $+0.000$]
Gemini-3.5	0				no eligible task			
Qwen3.8-Max	5	4/5	4/5	1/6	2/6	4/5	4/5	$+0.167$ [$+0.000$, $+0.600$]
DeepSeek-V4-Pro	11	0/13	1/13	1/21	6/21	0/11	0/11	$+0.238$ [$+0.000$, $+0.577$]
Qwen3.5-27B	4	1/4	1/4	1/4	1/4	1/4	1/4	$+0.000$ [$+0.000$, $+0.000$]
Gemma-4-31B	1	0/1	0/1	0/1	0/1	0/1	0/1	$+0.000$ [$+0.000$, $+0.000$]
All (API)†	43	16/45	17/45	5/55	10/55	15/43	16/43	$+0.091$ [-0.045, $+0.255$]

Table 9: Directive-removal control, per model (main-grid $K{=}2$ protocol). O = original system prompt and revision directive; N = the system prompt's last sentence removed and the directive reduced to "Revise the artifact. Return the complete revised artifact in a single fenced code block." Upper block: audio and SVG, pooled per model. n = eligible tasks of the model (round-0 draft with at least one exposed and one hidden defect at $K{=}2$); repair counts are round-0 defects repaired at the terminal state; false compl. = trajectories that end in false completion (Sec. 3). The last column is the defect-level hidden-repair difference, N minus O, with a task-bootstrap CI. †Pooled over the seven models with an eligible API-workflow task (43 (model, task) units over 24 tasks); the CI resamples the 24 tasks, each carrying all of its model units.

Length-matched report. Each exposed check adds one fixed-width line to the check report, so a $K{=}2$ report is six lines shorter than a $K{=}8$ report. To separate report length from the information it carries (robustness analysis (iii) of Sec. 5), the $K{=}2$ condition of the main grid was also run with every report padded to eight lines (the padded arm): the two real lines were followed by six content-free

lines of the same width, with dashes in all three fields of the line template `<family> <STATUS> | <witness>` (Appendix A), so they carry no check name, verdict, or counterexample, and no "not checked" notice. Everything else was the main grid's $K{=}2$ protocol: the same masks, the same cached round-0 drafts, reasoning off, temperature 0, seed 0, 16,384 output tokens, the all-clear stop and the five-round horizon, so a task whose draft already passed its exposed checks terminated at round 0 in both arms. All 1,200 (model, domain, task) units were run except one Gemma-4-31B audio task excluded as a truncated draft (1,199 pairs; 434 required at least one model call). Table 10 compares the padded arm with the main-grid $K{=}2$ arm task by task. Padding leaves hidden-defect repair unchanged and lowers the residual defect count by 0.027 $[0.004, 0.050]$, one twenty-sixth of the $K{=}2$-to-$K{=}8$ difference of 0.696; the change comes from repair of exposed defects ($+4.4$ pp $[+2.4, +6.5]$, concentrated in Qwen3.8-Max at $+15.8$ pp), which also lets slightly more trajectories reach an all-clear stop (1,048 vs. 1,029 of 1,199; mean rounds -0.05) and so ends slightly more of them in false completion (900 vs. 888; $+1.0$ pp of all trajectories $[+0.3, +1.8]$); conditional false completion (the false-completion rate among all-clear stops) is unchanged (86% in both arms).

		hidden repair			exposed repair		residual defects		
Model	n	padded	$K{=}2$	Δ (pp) [95% CI]	padded	$K{=}2$	padded	$K{=}2$	Δ [95% CI]
GPT-5.4	150	5/207	5/207	$+0.0$ $[+0.0, +0.0]$	49/68	45/68	1.49	1.53	-0.03 $[-0.07, +0.00]$
Sonnet-5	150	17/176	20/176	-1.7 $[-4.5, +0.0]$	55/60	54/60	1.13	1.10	$+0.03$ $[+0.01, +0.07]$
Haiku-4.5	150	11/229	10/229	$+0.4$ $[-0.9, +2.0]$	46/70	43/70	1.66	1.69	-0.03 $[-0.09, +0.02]$
Gemini-3.5	150	16/176	11/176	$+2.8$ $[+0.0, +8.9]$	36/54	34/54	1.19	1.24	-0.05 $[-0.15, +0.00]$
Qwen3.8-Max	150	17/206	15/206	$+1.0$ $[-1.4, +3.4]$	46/57	37/57	1.37	1.45	-0.08 $[-0.15, -0.02]$
DeepSeek-V4-Pro	150	14/258	13/258	$+0.4$ $[-0.8, +1.8]$	41/74	40/74	1.91	1.95	-0.04 $[-0.12, +0.01]$
Qwen3.5-27B	150	28/243	24/243	$+1.6$ $[-0.8, +5.4]$	41/69	39/69	1.71	1.71	$+0.00$ $[-0.11, +0.12]$
Gemma-4-31B	149	13/170	12/170	$+0.6$ $[+0.0, +2.0]$	22/48	22/48	1.23	1.23	-0.01 $[-0.03, +0.01]$
All	1199	121/1665	110/1665	$+0.7$ $[-0.2, +1.6]$	336/500	314/500	1.46	1.49	-0.03 $[-0.05, -0.00]$

Table 10: Length-matched $K{=}2$ arm (report padded to the $K{=}8$ line count) versus the main-grid $K{=}2$ arm, paired on task (same mask, same round-0 draft). Repair counts are round-0 defects repaired at the terminal state; Δ = padded minus $K{=}2$ with 95% task-bootstrap CIs ($B{=}10$k; the pooled row resamples domain/task clusters).

Fixed-horizon control for repair locality ($K{=}2$). It tests the claim in Sec. 5.1 that early stopping does not explain the gap. Under the all-clear stop a trajectory ends as soon as its exposed checks pass, so the exposed–hidden gap could reflect the rounds that early stopping cuts off. Here every eligible unit runs exactly three revision rounds at $K{=}2$ (no early stopping), with the main-grid $K{=}2$ masks and the shared round-0 drafts. The three-domain fixed-horizon control of Sec. 6.1 (Table 15) uses the same protocol at $K \in \{0, 2, 7\}$ on 10 tasks per domain; the units from those tasks are taken from its $K{=}2$ runs. Eligible units are (model, task) pairs whose shared round-0 draft has at least one exposed and one hidden defect under the $K{=}2$ mask: 279 units from the 50 audio and 50 SVG tasks, and 43 units from the 50 apiwf tasks. Estimator: for each unit, the exposed minus the hidden repair rate at round 3, averaged with equal weight over the model's audio and SVG units; task bootstrap $B{=}10$k, seed 0 (Table 11). The apiwf units are too few per model for per-model estimates (Gemini-3.5 has none; five models have five or fewer), so they are pooled over models, with a bootstrap that resamples the 24 tasks, each carrying all of its model units. Every apiwf unit fails `completeness` at round 0, and `completeness` is repaired in none of them, so the mean difference is -0.14 over the 17 units whose mask exposes `completeness` and $+0.58$ over the 26 that hide it (rows "compl. exposed" and "compl. hidden" of Table 11). Two DeepSeek-V4-Pro units whose round-0 reply contains no program, and which therefore fail all eight checks at round 0, supply 10 of the 14 hidden repairs; without them the pooled difference is $+0.32$ $[+0.13, +0.53]$.

Model	units (audio/svg)	exposed−hidden [95% CI]	exposed (pooled)	hidden (pooled)
GPT-5.4	33 (17/16)	$+0.67$ [$+0.51$, $+0.82$]	76%	11%
Sonnet-5	41 (24/17)	$+0.85$ [$+0.74$, $+0.94$]	98%	29%
Haiku-4.5	45 (30/15)	$+0.62$ [$+0.47$, $+0.76$]	72%	11%
Gemini-3.5	24 (14/10)	$+0.69$ [$+0.51$, $+0.85$]	74%	28%
Qwen3.8-Max	34 (21/13)	$+0.58$ [$+0.42$, $+0.74$]	73%	26%
DeepSeek-V4-Pro	40 (26/14)	$+0.57$ [$+0.42$, $+0.71$]	58%	4%
Qwen3.5-27B	39 (32/7)	$+0.56$ [$+0.42$, $+0.71$]	67%	23%
Gemma-4-31B	23 (10/13)	$+0.45$ [$+0.26$, $+0.64$]	55%	22%
API workflows[†]	43	$+0.30$ [$+0.12$, $+0.50$]	42%	25%
compl. exposed	17	-0.14 [-0.31, -0.02]	6%	32%
compl. hidden	26	$+0.58$ [$+0.42$, $+0.77$]	67%	21%

Table 11: Fixed-horizon control for repair locality: paired exposed-minus-hidden repair at a fixed three-round horizon ($K=2$; 279 eligible audio/svg and 43 apiwf (model, task) units). The estimate is the mean over units of the within-unit difference (exposed minus hidden repair rate at round 3). The last two columns pool defects instead: the share of all exposed, and of all hidden, round-0 defects of these units that pass at round 3, so their difference need not equal the estimate. All eight per-model (audio/svg) CIs exclude zero. [†]Pooled over seven models (Gemini-3.5 has no eligible unit); the CI resamples the 24 tasks, each carrying all of its model units. The two rows under "API workflows" split the apiwf units by whether `completeness` is exposed (compl. exposed) or hidden (compl. hidden).

	GPT-5.4	Sonnet-5	Haiku-4.5	Gemini-3.5	Qwen3.8-Max	DeepSeek-V4-Pro	Qwen3.5-27B	Gemma-4-31B
Same residual in all 3 runs	19/30	23/30	18/30	26/30	18/30	21/30	16/30	22/30
Largest spread (defects)	2	2	3	1	2	2	7	2
$K=2$/$K=8$ order reversed	1	0	2	0	0	0	0	0

Table 12: Rerun panel (robustness analysis (iv) of Sec. 5): 15 tasks per model (5 per domain) at $K \in \{2, 8\}$, run three times: the main-grid run and two reruns, each rerun with its own round-0 draft shared by its $K=2$ and $K=8$ arms. Row 1: (task, K) pairs, of 30, whose terminal residual defect count is the same in all three runs. Row 2: the largest spread (maximum minus minimum over the three runs) of that count. Row 3: tasks, of 15, whose $K=2$ and $K=8$ residual counts are ordered one way in one run and the opposite way in another. The Qwen3.5-27B spread of 7 comes from one run whose round-0 draft was truncated.

Within-check comparison. It tests the claim in Sec. 5.1 that exposed checks are not inherently easier to repair. Because masks differ across tasks, each check is exposed in some tasks and hidden in others; the exposed-minus-hidden repair difference computed within each check and averaged with equal check weights (over the three to six checks per cell with both exposed and hidden defects; parseable trajectories only, as for P1) at $K=2$ is positive in all 16 audio/svg cells ($+0.39$ to $+0.99$); in apiwf, one to three checks per cell have both kinds of defect: `completeness` in every cell, whose defects are almost never repaired whether exposed or hidden (exposed: none of 11–12 in seven cells and 3 of 8 for Sonnet-5; hidden: none of 29–38 in any cell), and at most two others, most with only one or two defects on one side, so the comparison is not informative there.

Further descriptives. Conditional false completion at $K=2$ (Fig. 2d) is 40–100% in audio (Gemma-4-31B 40%, 17 of its 42 all-clear stops; the other seven models 84–100%), 65–87% in svg, and 69–100% in apiwf (100% for five models). Repair at $K=8$ of the defects in the drafts falsely cleared at $K=2$ (Fig. 2c) is 50–100% per model in audio, 55–97% in svg, and 4–17% in apiwf except Sonnet-5 (56%). Of the 888 false completions at $K=2$ (Fig. 2e), 668 occur at round 0 and 220 after revisions: 183 and 125 in audio, 217 and 78 in svg, and 268 and 17 in apiwf. Fate of round-0 defects at $K=2$, at the terminal state (eight models pooled): in audio, of 1,002 round-0 defects, 255 are repaired and 747 still fail, and 36 checks that passed at round 0 fail at the terminal state; in svg, 130 and 552 of 682, with 3 such newly failing checks; in apiwf, 39 and 450 of 489, with 5. Cumulative repair of round-0 defects by round (pooled over models; share repaired after round 1 $\rightarrow$ after round 5, a trajectory that stopped earlier keeping its terminal state): audio $K=2$

18→25% vs. K=8 34→68%; svg 15→19% vs. 51→79%; apiwf 7→8% vs. 11→18% (in apiwf, both level off by round 3). Truncation (Appendix G): 95 of 6,000 main-grid trajectories (1.6%) end truncated at the output cap without the model being told, concentrated in audio.

API-workflow check granularity. These analyses support Sec. 5.3, where repair locality is weaker in API workflows than in audio and SVG. The tasks are solvable: every apiwf task has a reference program that passes all eight checks in the same sandbox; 36 of the 50 tasks were made fully correct by at least one model in some condition (31 at K=8), and for one task that no model made fully correct at K=8, two small hand edits to Qwen3.8-Max's own failing draft make it pass all eight. Checks differ in scope (Sec. 4). In apiwf, `completeness` (AppWorld's task-outcome assertion suite) accounts for 80% of round-0 defects, fails in 95% of drafts, and is the check that reporting helps least (Table 13); in audio and svg the largest check accounts for 42% and 31% of round-0 defects, and exposed defects of every check that has round-0 defects are repaired at 33–100%. Splitting the apiwf trajectories at $K < 8$ by whether `completeness` is in the mask (Table 14): with it hidden, the loop reaches an all-clear stop in 626 of 655 trajectories and 95% of those stops are false completions; with it exposed, it reaches an all-clear stop in 89 of 937 trajectories and 3% of those stops (3 of 89) are false completions, and P1 is -0.11 because most exposed defects are then `completeness` failures, which are rarely repaired (Table 13). Answer-only failures (`completeness` failing only its "assert answers match" assertion, world state correct) are 22 of 380 round-0 completeness failures, 106 of 1,830 terminal ones, and the sole remaining failure in 43 of 599 false completions (7%); counting them as fully correct raises the number of fully correct K=8 trajectories from 38 to 56 of 398 (GPT-5.4 2→8, Sonnet-5 29→31, Gemini-3.5 1→2, Qwen3.8-Max 5→6, DeepSeek-V4-Pro 0→1, Qwen3.5-27B 0→4, Gemma-4-31B 1→4, Haiku-4.5 0→0). The eight apiwf P1 values follow, each with its 95% task-bootstrap interval and, after the arrow, the interval obtained by resampling the ten AppWorld task families, of which the 50 tasks are instances, instead of the tasks (B=10,000); tasks of one family share its world and instruction wording (Appendix A). Family resampling widens six of the eight intervals, narrows those of Haiku-4.5 and Gemini-3.5 slightly, and leaves the same six of eight (unadjusted) intervals excluding zero: GPT-5.4 $+0.21$ $[+0.13, +0.29]$→$[+0.08, +0.35]$; Sonnet-5 $+0.38$ $[+0.21, +0.52]$→$[+0.16, +0.57]$; Haiku-4.5 $+0.13$ $[+0.01, +0.23]$→$[+0.03, +0.23]$; Qwen3.8-Max $+0.19$ $[+0.09, +0.29]$→$[+0.05, +0.30]$; DeepSeek-V4-Pro $+0.03$ $[+0.01, +0.06]$→$[+0.01, +0.07]$; Qwen3.5-27B $+0.11$ $[+0.04, +0.18]$→$[+0.02, +0.20]$; Gemini-3.5 0.00 $[-0.10, +0.07]$→$[-0.09, +0.06]$; Gemma-4-31B $+0.04$ $[-0.06, +0.12]$→$[-0.04, +0.18]$. With one-sided tests for a positive P1, 21 of the 24 model–domain comparisons are significant after Holm correction; the weakest audio/svg cell, Gemma-4-31B on audio, has adjusted p=0.0055. The exceptions are Haiku-4.5, Gemma-4-31B, and Gemini-3.5 on API workflows (adjusted p=0.060, 0.437, and 0.458, respectively); Haiku-4.5's interval is thus one of the six unadjusted intervals above that exclude zero, but its P1 is not significant after Holm correction.

apiwf check	share of round-0 defects	drafts failing	repaired when exposed (n)	repaired when hidden (n)
completeness	0.80	0.95	0.05 (895)	<0.01 (625)
argument_fidelity	0.13	0.15	0.51 (154)	0.05 (86)
authentication_first	0.04	0.05	0.98 (48)	0.31 (36)
dependency_order	0.01	0.02	0.42 (19)	0.00 (9)
exactly_once	0.01	0.01	1.00 (6)	0.00 (6)
side_effect_conservation	<0.01	<0.01	0.50 (4)	— (0)
entity_binding	<0.01	<0.01	1.00 (2)	0.00 (2)
precondition_reads	0	0	—	—

Table 13: API-workflow checks in the main grid (eight models; 398 parseable drafts): share of all round-0 defects, share of drafts failing the check, and the repair rate of round-0 defects when the check is exposed vs. hidden ($K \in \{2, 4, 6, 7\}$, pooled over models). A draft's defect enters once per K, so n counts trajectories: each failing draft's four K values split between the two columns (for `completeness`, $895 + 625 = 4 \times 380$).

apiwf trajectories, $K \in \{2, 4, 6, 7\}$	completeness hidden	completeness exposed
trajectories	655	937
all-clear stops	626	89
conditional false completion	95%	3%
P1 (repair locality)	$+0.63$	-0.11
terminal residual defects (mean)	1.13	1.04
hidden repair in the other seven checks	1%	24%

Table 14: The apiwf trajectories of Table 13 (parseable round-0 and terminal states; 1,592 of 1,600) split by whether `completeness` is in the mask. Conditional false completion is the share of all-clear stops that are false completions; the last row is the terminal repair rate of hidden round-0 defects in checks other than `completeness`.

C RQ2 SUPPLEMENTS

This appendix supports Sec. 6. Table 15 and Fig. 6a detail the three-domain fixed-horizon control (Sec. 6.1), and the paragraph between them explains what U_K and the stopping component measure. Fig. 6b and Table 16 give, per cell, the comparison of $K{=}2$, the GS policy, and $K{=}8$ on the same round-0 drafts (Sec. 6.3). The remaining paragraphs and Tables 17–19 give the design, inference, and full results of the matched continuation control (Sec. 6.2).

Model	floor ($K{=}0$)	net repairs $K7{-}K2$ (defects/task) [95% CI]	hidden@$K2$	exposed@$K2$	stopping comp. [95% CI] (n)
GPT-5.4	8/59	$+0.23$ [-0.30, $+0.67$]	8/42 (19%)	10/17 (59%)	$+0.04$ [$+0.00$, $+0.10$] (27)
Sonnet-5	12/46	$+0.43$ [$+0.13$, $+0.73$]*	11/33 (33%)	11/13 (85%)	$+0.16$ [$+0.02$, $+0.32$]* (22)
Haiku-4.5	4/58	$+0.50$ [$+0.13$, $+0.87$]*	5/42 (12%)	10/16 (63%)	$+0.13$ [$+0.03$, $+0.26$]* (24)
Gemini-3.5	3/37	$+0.40$ [$+0.20$, $+0.60$]*	2/26 (8%)	6/11 (55%)	$+0.09$ [$+0.00$, $+0.23$] (22)
Qwen3.8-Max	10/49	$+0.10$ [-0.43, $+0.53$]	7/36 (19%)	7/13 (54%)	$+0.01$ [$+0.00$, $+0.04$] (21)
DeepSeek-V4-Pro	7/60	$+0.20$ [$+0.00$, $+0.43$]	9/44 (20%)	7/16 (44%)	$+0.07$ [$+0.01$, $+0.14$]* (24)
Qwen3.5-27B	13/66	$+0.53$ [$+0.33$, $+0.77$]*	10/49 (20%)	9/17 (53%)	$+0.02$ [$+0.00$, $+0.05$] (28)
Gemma-4-31B	5/43	$+0.37$ [$+0.17$, $+0.60$]*	7/31 (23%)	6/12 (50%)	$+0.03$ [$+0.00$, $+0.08$] (22)

Table 15: Three-domain fixed-horizon control of Sec. 6.1: exactly three revision rounds at every $K \in \{0, 2, 7\}$, no all-clear stop, 10 tasks per domain, main-grid round-0 drafts and masks. At $K{=}0$ there is no check report: in every round the model receives only the bare revision instruction *"Revise the artifact. Return the complete revised artifact in a single fenced code block."* At $K{=}2$ and $K{=}7$ it receives this instruction alone in rounds where every exposed check passes, and otherwise the check report followed by the revision directive. Floor = round-0 defects repaired by round 3 at $K{=}0$ (repaired/total; 7–26%). Net repairs $K7{-}K2$ = per-task difference between $K{=}7$ and $K{=}2$ in the number of round-0 defects net repaired by round 3 (checks failing at round 0 minus checks failing at round 3), averaged over the 30 tasks of the three domains; the CI is a percentile task bootstrap ($B{=}10$k). It is a difference of defect counts per task (Sec. 6.1), not of proportions, so it is not bounded by 1. Hidden/exposed = repair of round-0 defects at $K{=}2$ under the fixed horizon (repaired/total). Stopping component = hidden repair at $K{=}2$ under this control minus that in the main grid (all-clear stop) on identical drafts, paired over the n tasks with hidden round-0 defects (task bootstrap $B{=}10$k). * = CI excludes zero.

What U_K and the stopping component measure. U_K (Sec. 5.2) is computed from the shared round-0 drafts and the fixed masks alone, so it contains no revision behavior by construction: holding the drafts fixed and changing only which checks are labeled exposed (the $K{=}2$ versus $K{=}7$ masks) gives exactly the $U_2 - U_7$ values of Table 8 ($+0.22$ to $+0.56$ over the 16 audio/svg cells). Revision can change false completion only through all-clear stops after round 0. The stopping component measures how much hidden repair early stopping forgoes (Sec. 6.1): the paired contrast in hidden repair at $K{=}2$ between the three-domain fixed-horizon control and the main grid, on identical drafts, puts it at $+0.01$ to $+0.16$ (CI excludes zero for 3/8 models; Table 15).

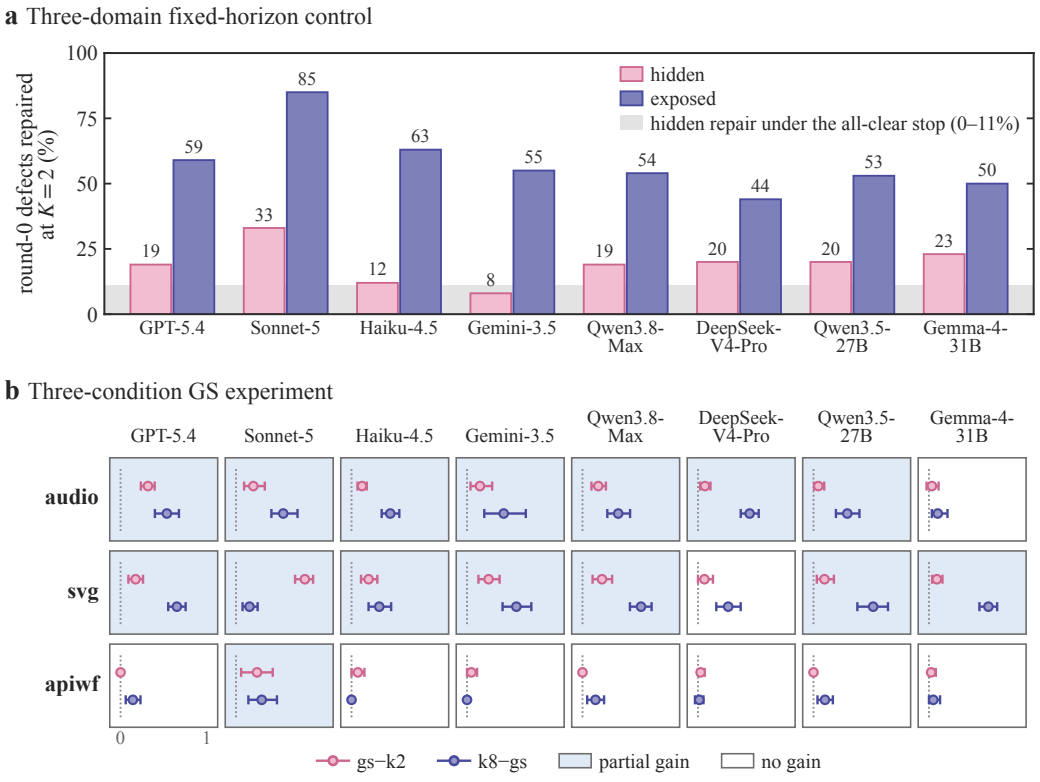

Figure 6: Three-domain fixed-horizon control and three-condition global-stop experiment. (a) Three-domain fixed-horizon control: hidden- vs. exposed-defect repair of round-0 defects at $K{=}2$ under a forced three-round horizon (10 tasks per domain; the hidden@$K2$ and exposed@$K2$ columns of Table 15); gray band = range of $K{=}2$ hidden repair under the all-clear stop across the 16 audio/svg main-grid cells (0–11%; Table 7). (b) Three-condition global-stop experiment (all main-grid tasks; $K{=}2$ / global-stop / $K{=}8$): per-cell paired hidden-repair steps gs−k2 (GS minus $K{=}2$) and k8−gs ($K{=}8$ minus GS), and each cell's label: *partial gain* if GS beats $K{=}2$ and $K{=}8$ still beats GS (both CIs exclude zero), *no gain* if the gs−k2 CI includes zero (full rule in the caption of Table 16); every cell carries the same x-axis (0 to 1: the step in hidden-defect repair rate, i.e., the change in the share of $K{=}2$-hidden round-0 defects repaired at the terminal state; dotted line marks zero). False completion is zero under global-stop by construction (stopping rule), not through repair.

Domain	Model	gs−k2 hid. repair Δ	gs−k2 hid. repair 95% CI	k8−gs hid. repair Δ	k8−gs hid. repair 95% CI	$K2$/GS/$K8$ fully correct	$K2$/GS/$K8$ rounds	gain
audio	GPT-5.4	+.317	[.235,.395]	+.537	[.398,.676]	.12/.16/.88	1.0/4.7/3.2	partial
audio	Sonnet-5	+.200	[.093,.333]	+.547	[.410,.712]	.04/.24/.96	1.0/4.7/2.9	partial
audio	Haiku-4.5	+.121	[.069,.176]	+.449	[.349,.553]	.04/.04/.32	1.3/4.9/4.3	partial
audio	Gemini-3.5	+.150	[.040,.291]	+.425	[.200,.681]	.14/.26/.82	1.0/4.3/2.4	partial
audio	Qwen3.8-Max	+.183	[.104,.271]	+.413	[.287,.551]	.04/.10/.56	1.1/4.8/4.1	partial
audio	DeepSeek-V4-Pro	+.080	[.025,.144]	+.598	[.495,.703]	.02/.04/.46	1.4/4.9/4.6	partial
audio	Qwen3.5-27B	+.056	[.007,.124]	+.392	[.260,.532]	.00/.00/.10	1.9/5.0/4.7	partial
audio	Gemma-4-31B	+.034	[−.028,.111]	+.102	[.033,.214]	.50/.52/.64	0.9/2.4/2.1	none
svg	GPT-5.4	+.174	[.091,.260]	+.652	[.552,.754]	.12/.22/.74	1.0/4.1/2.9	partial
svg	Sonnet-5	+.797	[.684,.892]	+.156	[.076,.250]	.20/.76/.96	0.4/2.9/1.6	partial
svg	Haiku-4.5	+.198	[.110,.299]	+.321	[.197,.459]	.14/.18/.50	1.2/4.4/3.4	partial
svg	Gemini-3.5	+.250	[.132,.377]	+.571	[.412,.745]	.26/.38/.86	0.4/3.5/1.8	partial
svg	Qwen3.8-Max	+.226	[.117,.343]	+.677	[.549,.800]	.22/.30/.84	0.6/3.7/2.4	partial
svg	DeepSeek-V4-Pro	+.075	[.000,.173]	+.350	[.212,.494]	.12/.16/.42	1.1/4.4/3.4	none
svg	Qwen3.5-27B	+.127	[.038,.236]	+.691	[.508,.863]	.34/.40/.92	0.3/3.3/1.8	partial
svg	Gemma-4-31B	+.091	[.036,.156]	+.688	[.584,.789]	.14/.18/.70	0.8/4.3/3.0	partial
apiwf	GPT-5.4	+.000	[.000,.000]	+.143	[.060,.231]	.00/.00/.04	1.8/5.0/5.0	none
apiwf	Sonnet-5	+.243	[.059,.425]	+.297	[.143,.474]	.28/.36/.58	0.8/3.5/2.8	partial
apiwf	Haiku-4.5	+.073	[.000,.149]	+.000	[.000,.000]	.00/.00/.00	1.3/5.0/5.0	none
apiwf	Gemini-3.5	+.050	[.000,.116]	+.000	[.000,.000]	.02/.02/.02	1.1/4.9/4.9	none
apiwf	Qwen3.8-Max	+.000	[.000,.000]	+.150	[.054,.250]	.08/.08/.10	1.3/4.6/4.6	none
apiwf	DeepSeek-V4-Pro	+.030	[.000,.077]	+.015	[−.035,.063]	.00/.00/.00	1.6/5.0/5.0	none
apiwf	Qwen3.5-27B	+.000	[.000,.000]	+.133	[.047,.224]	.00/.00/.00	1.2/5.0/5.0	none
apiwf	Gemma-4-31B	+.025	[.000,.079]	+.050	[.000,.128]	.00/.00/.02	1.2/5.0/4.9	none

Table 16: Full global-stop table, all 24 model–domain cells (gs−k2 = GS minus $K{=}2$; k8−gs = $K{=}8$ minus GS; hid. = the checks that the $K{=}2$ mask hides): paired hidden-repair effects (task bootstrap $B{=}10$k) with per-condition fully correct rates and mean revision rounds. Gain column: *none* if the gs−k2 CI includes zero; *harmful* if it lies below zero; otherwise *partial* if the k8−gs CI excludes zero and *full* if it includes zero. Result: 15 partial, 9 none, 0 full, 0 harmful; gs−k2 on exposed-defect repair includes zero in all 24 cells (-0.08 to $+0.17$). The output-truncation sensitivity analysis (excluding tasks with any truncated call in any condition) moves one of the 24 cells (Qwen3.5-27B, audio: partial gain to no gain). In the GS condition every round-0 draft was revised at least once, including the 97 drafts that already passed all eight checks and that the GS stopping rule (stop once all eight checks pass) would have accepted at round 0; these drafts received the all-pass $K{=}2$ report and the revision directive, but not the "not fully correct" message, which is appended only when a hidden check fails. After that first revision, 15 of the 97 were no longer fully correct, and 9 were still not fully correct at the terminal state. The fully correct and rounds columns score those drafts as round-0 stops, as the GS stopping rule specifies. Hidden-repair effects are unaffected, since such drafts carry no defects. The 11 false-clear states that descend from these drafts are described with the matched continuation control below.

Matched continuation control: design details. The 904 false-clear states are the first false-clear states (the exposed $K{=}2$ checks pass while a hidden check fails) of the global-stop trajectories (Table 17 gives their number per model and domain); the other 296 of the 1,200 model×domain×task units never reach such a state, so they are never forked and receive neither sentence. These states carry 1,399 hidden defects in total, the *trigger-time hidden defects* (the hidden defects failing at the fork in Sec. 6.2; every check failing at such a state is hidden, since all exposed checks pass by definition). By the round of the first post-fork revision (the first revision whose prompt carries a branch's sentence), the states split as round 1: 668 states / round 2: 151 / round 3: 52 / round 4: 23 / round 5: 10; the last group has exactly one post-fork revision, so its one-step and terminal outcomes coincide. The 668 states whose first post-fork revision is round 1 are exactly the round-0 drafts falsely cleared at $K{=}2$ in the main grid (the U_2 values of Tables 7 and 8, times 50 tasks, sum to 668); the other 236 arise after revisions. Eleven of these 236, all in the round-2 group, exist only because every round-0 draft was revised at least once in the GS condition (caption of Table 16): each is the round-1 artifact of one of the 97 drafts that already passed all eight checks. All estimates below include them; excluding them moves the one-step and terminal estimates by 0.1 and 0.2 pp. After the fork, each branch's $K{=}2$ report is recomputed every round from its own artifact, its sentence is re-appended at

every later false-clear state, each branch stops only when all eight checks pass or after round 5 (the GS stopping rule), and the five-round horizon counts from the trajectory's original start, not from the fork. The continuation messages are, verbatim, Neutral: "Another revision round is now available. Please continue revising."; GS (the "not fully correct" message): "The artifact is not fully correct yet. Please continue revising."

Model	audio	svg	apiwf	total
GPT-5.4	43	42	33	118
Sonnet-5	48	40	32	120
Haiku-4.5	42	35	38	115
Gemini-3.5	39	37	38	114
Qwen3.8-Max	44	39	34	117
DeepSeek-V4-Pro	43	35	34	112
Qwen3.5-27B	38	32	38	108
Gemma-4-31B	23	39	38	100
total	320	299	285	904

Table 17: False-clear states per model and domain.

Estimands and inference. Let D_s be the set of checks failing at false-clear state s, i.e., its trigger-time hidden defects ($\sum_s |D_s| = 1{,}399$). The primary (one-step) outcome pools defects: $P^b = \sum_s \#\{f \in D_s \text{ passing after the first post-fork revision in branch } b\} / \sum_s |D_s|$, and the primary estimand is the paired difference $\theta_1 = P^{\mathrm{GS}} - P^{\mathrm{Neutral}}$ ($+0.036$ $[+0.024, +0.048]$, $p < 0.001$; the one-step gain of Sec. 6.2); the terminal estimand θ_H is identical with the first post-fork revision replaced by the branch's terminal state (all eight checks passing stops the branch, else round 5; $\theta_H = +0.090$ $[+0.070, +0.112]$, the terminal gain of Sec. 6.2). In counts, the GS and Neutral branches repair 132 and 82 of the 1,399 trigger-time hidden defects one step after the fork (9.4% and 5.9%), and 253 and 127 by the terminal state (18.1% and 9.1%). Inference hierarchy: pooled one-step primary; pooled terminal secondary; three domain effects (Table 18) with Holm correction over the three domains within outcome; 24 model×domain cells (Table 19) uncorrected; policy-level estimands over all 1,200 units (next paragraph) secondary. Confidence intervals: percentile bootstrap, B=10,000, seed 0, with two-sided p-values from the same resamples (twice the smaller share of resampled differences on either side of zero); pooled analyses resample the 138 domain×task clusters that contain at least one false-clear state (a task appears under up to eight models), domain analyses resample tasks within the domain, and cell analyses resample paired tasks within the cell, always preserving the GS/Neutral pairing. A state-equal-weighted re-estimate (each state's within-state repair share weighted equally) gives $\theta_1 = +0.030$ $[+0.020, +0.042]$ and $\theta_H = +0.080$ $[+0.060, +0.102]$.

Outcome	Domain	Δ	95% CI	Holm p
one-step	audio	$+0.035$	$[+0.016, +0.054]$	.001*
one-step	svg	$+0.043$	$[+0.021, +0.064]$	.001*
one-step	apiwf	$+0.027$	$[+0.006, +0.052]$	.003*
terminal	audio	$+0.090$	$[+0.064, +0.118]$	.001*
terminal	svg	$+0.136$	$[+0.093, +0.184]$	.001*
terminal	apiwf	$+0.021$	$[+0.000, +0.046]$	.063

Table 18: Matched-continuation domain-level paired effects (secondary): Δ = GS branch minus Neutral branch in the share of trigger-time hidden defects repaired, one-step or terminal (Holm correction over the three domains within each outcome; * = significant after correction).

Domain	Model	one-step Δ	one-step 95% CI	terminal Δ	terminal 95% CI
audio	GPT-5.4	+.067	[+.015, +.123]	+.120	[+.041, +.203]
audio	Sonnet-5	+.049	[+.000, +.106]	+.164	[+.075, +.255]
audio	Haiku-4.5	+.040	[+.000, +.090]	+.110	[+.058, +.168]
audio	Gemini-3.5	+.020	[+.000, +.065]	+.180	[+.082, +.286]
audio	Qwen3.8-Max	+.000	[−.074, +.075]	+.012	[−.073, +.098]
audio	DeepSeek-V4-Pro	+.072	[+.021, +.135]	+.082	[+.030, +.146]
audio	Qwen3.5-27B	+.000	[+.000, +.000]	+.046	[+.011, +.094]
audio	Gemma-4-31B	+.000	[+.000, +.000]	+.000	[+.000, +.000]
svg	GPT-5.4	+.029	[+.000, +.069]	+.074	[−.031, +.181]
svg	Sonnet-5	+.143	[+.044, +.246]	+.556	[+.414, +.695]
svg	Haiku-4.5	+.033	[−.033, +.097]	+.000	[−.123, +.115]
svg	Gemini-3.5	+.137	[+.043, +.250]	+.294	[+.163, +.426]
svg	Qwen3.8-Max	−.016	[−.068, +.034]	+.078	[−.029, +.190]
svg	DeepSeek-V4-Pro	+.000	[+.000, +.000]	+.106	[+.000, +.231]
svg	Qwen3.5-27B	+.043	[+.000, +.102]	−.021	[−.122, +.068]
svg	Gemma-4-31B	+.000	[−.041, +.042]	+.014	[−.030, +.065]
apiwf	GPT-5.4	+.064	[+.000, +.137]	+.000	[+.000, +.000]
apiwf	Sonnet-5	+.053	[+.000, +.158]	+.105	[+.000, +.237]
apiwf	Haiku-4.5	+.024	[+.000, +.073]	+.024	[+.000, +.073]
apiwf	Gemini-3.5	+.050	[+.000, +.116]	+.050	[+.000, +.116]
apiwf	Qwen3.8-Max	+.000	[+.000, +.000]	−.027	[−.086, +.000]
apiwf	DeepSeek-V4-Pro	+.022	[+.000, +.068]	+.022	[+.000, +.067]
apiwf	Qwen3.5-27B	+.000	[+.000, +.000]	+.000	[+.000, +.000]
apiwf	Gemma-4-31B	+.000	[+.000, +.000]	+.000	[+.000, +.000]

Table 19: Matched-continuation model×domain paired effects (Δ as in Table 18; uncorrected). Entries with $[+.000, +.000]$ are exact ties: at that outcome, every false-clear state of the cell has the same number of trigger-time hidden defects repaired in both branches.

Supporting outcomes and policy-level estimands. Supporting outcomes on the 904 false-clear states (GS branch minus Neutral branch; residual defects counted per state): one-step fully correct rate $+0.013$ $[+0.004, +0.023]$; one-step residual defects $+0.006$ $[-0.038, +0.052]$ (not significant; a CI covering zero is read as no detectable change, not as equivalence); terminal fully correct rate $+0.059$ $[+0.041, +0.078]$; terminal residual defects -0.096 $[-0.142, -0.052]$. In counts, 81 GS branches vs. 28 Neutral branches became fully correct within the horizon (81/904 and 28/904 are the terminal fully correct rates; mean post-fork revisions among those that did: 2.48 vs. 2.04); the remaining 823 GS and 876 Neutral branches reached round 5 without becoming fully correct. Policy-level estimands are the same GS-minus-Neutral contrasts averaged over all 1,200 model×domain×task units run under the GS policy. The 296 units that never reach a false-clear state contribute an exactly zero contrast and stay in the denominator, so each point estimate is the state-level estimate above multiplied by $904/1{,}200$, and each CI is a state-level bootstrap interval multiplied by the same factor. With the "not fully correct" message, the fully correct rate is higher by $+0.010$ $[+0.003, +0.017]$ one step after the fork and by $+0.044$ $[+0.031, +0.059]$ at the terminal state, i.e., in the share of trajectories that end fully correct; residual defects change by $+0.004$ $[-0.027, +0.039]$ one-step and -0.073 $[-0.107, -0.040]$ terminal.

Execution order and completeness. Each GS branch was run immediately before its paired Neutral branch. All 1,808 branches (904 pairs) completed and entered the analysis; no branch was excluded. The endpoints used here return no immutable model-snapshot identifier; served-model strings and decoding parameters were identical within every pair.

D RQ3 SUPPLEMENTS AND DIAGNOSTICS

This appendix gives the design details and full results behind Sec. 7, namely the reasoning two-arm study and the fixed-draft repair study (Sec. 7.1) and the self-selection study (Sec. 7.2); it ends with API-workflow probe diagnostics that bear on Sec. 5.3.

Reasoning two-arm study: policies, tasks, and the interaction regression. The two-arm study of Sec. 7.1 runs the loop at all five coverage levels twice for each model and task, once with the model's reasoning mode on (the ON arm) and once with it off (the OFF arm): 8 models $\times$ 45 tasks (15 per domain) $\times$ 5 K values $=$ 1,800 on/off pairs, or 3,600 trajectories. The ON arm uses each model's built-in reasoning mode with `max_tokens` 32,768, which the reasoning of the two Qwen models can exceed (Appendix A): GPT-5.4 `reasoning_effort=medium`; Sonnet-5 adaptive thinking; Haiku-4.5 extended thinking with a 16,384-token budget (recorded reasoning exceeded it on 2 of 725 call records); Gemini-3.5 default dynamic thinking; Qwen3.8-Max thinking enabled without a budget; DeepSeek-V4-Pro `reasoning_effort=high`; Qwen3.5-27B and Gemma-4-31B chat-template thinking enabled. The OFF arm is the matched reasoning-disabled control with the same `max_tokens`, as in the fixed-draft repair study below. Arms share tasks, masks, and the all-clear stopping rule. One round-0 draft is drawn per (model, task, arm) and shared by that arm's five trajectories, one per K value; the two arms draw their own drafts, so the reasoning arm's draft is itself produced with reasoning on. Tasks are the first 15 positions of each domain's fixed task sequence, a fixed ordering of the domain's 50 tasks whose first 10 tasks form the three-domain fixed-horizon control (Appendix C) and whose first 5 form the rerun panel (Table 12). The interaction regression fits, over all 3,600 two-arm trajectories, $y = \beta_0 + \beta_1 x + \beta_2\,\mathrm{on} + \beta_3\, x \cdot \mathrm{on}$ by OLS, where y is the terminal residual defect count of a (model, domain, task, K) trajectory, $x = K/8$, and on indicates the reasoning arm; 95% CIs come from B=10,000 bootstrap resamples that draw the 15 tasks of each domain with replacement and carry, for every drawn task, all of its trajectories across the eight models, both arms, and the five K values (the domain$\times$task clusters of Appendix C), seed 0; the per-model estimates in Table 21 resample tasks within that model. The reasoning-on slope is $\beta_1+\beta_3 = -0.43$ $[-0.51, -0.34]$, its CI taken from the same resamples. A head-to-head comparison asks whether reasoning can stand in for coverage: within each of the 360 (model, domain, task) triplets, K=2 with reasoning *wins* if it ends with fewer terminal residual defects than K=8 without reasoning and *loses* if it ends with more: 82 wins, 76 losses, and 202 ties. With reasoning on, conditional false completion at K=2 is lower than in the OFF arm for every model (for example, Qwen3.8-Max: 23% of all-clear stops, against 78% in the OFF arm) and reaches zero for none. In API workflows, Qwen3.8-Max's reasoning, enabled without a budget, brings residual defects near zero ($\Delta = -0.83$ $[-0.96, -0.67]$; 60 of its 75 (task, K) pairs improved) and DeepSeek-V4-Pro's reduces them by 0.27 $[0.04, 0.48]$; the other six models show no detectable effect (Table 20). Median reasoning tokens per ON-arm call record (a shared draft's call counted once in each of its five K trajectories): GPT-5.4 2,008, Haiku-4.5 1,731, Qwen3.5-27B 766, Gemma-4-31B 3,269, Gemini-3.5 10,914, DeepSeek-V4-Pro 11,974, Qwen3.8-Max 23,471; for Sonnet-5 the median is 0 (nonzero on a minority of calls, maximum 22,902). These volumes are not comparable across models, since they confound the reasoning policy with tokenizer granularity; Table 20 reports each model's effect separately.

Model	audio Δ(on$-$off)	svg Δ	apiwf Δ
GPT-5.4	-0.56 $[-0.72, -0.41]^*$	-0.60 $[-0.87, -0.33]^*$	-0.17 $[-0.39, +0.04]$
Sonnet-5	-0.44 $[-0.61, -0.29]^*$	-0.44 $[-0.75, -0.20]^*$	-0.17 $[-0.41, +0.08]$
Haiku-4.5	-1.67 $[-2.17, -1.17]^*$	-0.45 $[-0.85, -0.12]^*$	$+0.01$ $[-0.07, +0.11]$
Gemini-3.5	-0.23 $[-0.41, -0.04]^*$	-0.08 $[-0.41, +0.33]$	$+0.28$ $[-0.03, +0.72]$
Qwen3.8-Max	-0.93 $[-1.24, -0.61]^*$	-0.49 $[-0.76, -0.23]^*$	-0.83 $[-0.96, -0.67]^*$
DeepSeek-V4-Pro	-1.65 $[-2.68, -1.01]^*$	-1.05 $[-1.67, -0.49]^*$	-0.27 $[-0.48, -0.04]^*$
Qwen3.5-27B	-1.61 $[-2.09, -1.11]^*$	-0.17 $[-0.49, +0.13]$	-0.01 $[-0.09, +0.07]$
Gemma-4-31B	-0.29 $[-0.56, -0.04]^*$	-0.49 $[-0.72, -0.27]^*$	-0.01 $[-0.04, +0.00]$

Table 20: Reasoning two-arm study: paired ON$-$OFF differences in the terminal residual defect count (negative $=$ fewer defects with reasoning); each entry averages one model's 75 (task, K) pairs in one domain (225 pairs per model); task bootstrap B=10k; * = CI excludes zero, uncorrected. A trajectory whose final artifact is invalid, for example because truncation cut it off, fails all eight checks and so counts as eight residual defects (Appendix A). Reasoning arms with truncated calls: Gemini-3.5 45/597 call records (in 30 trajectories: 16 apiwf, 13 svg, 1 audio) and DeepSeek-V4-Pro 56/616 (in 45 trajectories, all in svg); no other arm has a truncated call. Excluding every (task, K) pair in which either arm's trajectory has a truncated call leaves every conclusion unchanged (Gemini-3.5 audio -0.24 $[-0.42, -0.08]$, svg -0.23 $[-0.49, +0.05]$; DeepSeek-V4-Pro svg -0.90 $[-1.50, -0.30]$ on the remaining 30 pairs).

Draft quality and the interaction. This paragraph asks how much of the smaller slope with reasoning (Sec. 7.1) comes from cleaner drafts: reasoning-on drafts start with 1.09 defects on average, against 1.71 without (last column of Table 21). Because the two arms draw their own round-0 drafts, the draft's defect count can be adjusted for. Table 21 reports, for the pooled data, each domain, and each model: the slopes and interaction of the regression above, whose outcome is the terminal residual defect count (the *count* scale), over all 3,600 trajectories; the interaction from the same regression with the trajectory's own round-0 defect count r_0 and $r_0 \cdot x$ added as covariates (an analysis of covariance, column ANCOVA), over the 3,500 trajectories whose draft was parseable (an artifact could be extracted from it); and, over trajectories with $r_0 \geq 1$ and parseable drafts, the slopes and interaction on two per-defect scales: the *normalized residual defect count*, the terminal residual count divided by r_0 (Table 21; this ratio also counts defects introduced during repair and can exceed 1), and the *survival of round-0 defects*, the number of the draft's own defects still failing at the terminal state divided by r_0 (Table 22). The two scales differ only by defects introduced during repair, which are rare: 0.03 per reasoning-on and 0.06 per reasoning-off trajectory with a parseable draft, and any at all in 1.9% and 5.2% of those trajectories. Unparseable drafts (a response with no extractable artifact, scored as eight defects) are set aside for the adjusted estimators because they carry $r_0 = 8$ regardless of the model's ability. Fifteen reasoning-on and five reasoning-off drafts are unparseable, and since each is shared by five trajectories, 75 of 1,800 reasoning-on and 25 of 1,800 reasoning-off trajectories are affected, 45 of them DeepSeek-V4-Pro on svg (its reasoning output hit the 32,768-token allowance). CIs: the same domain×task cluster bootstrap as the main regression for the pooled and domain rows (B=10,000, seed 0); the model rows resample tasks within that model. Pooled normalized-residual slopes are -0.46 [-0.53, -0.40] without and -0.58 [-0.68, -0.48] with reasoning; on the count scale, the apiwf slope with reasoning is -0.29 [-0.43, -0.18].

	count (all trajectories)			ANCOVA	normalized residual (parseable)			round-0 defects
	off	on	interaction	interaction	off	on	interaction	off / on (unparseable)
All	-0.84	-0.43	+0.41 [+0.26, +0.58]	−0.06 [-0.19, +0.06]	-0.46	-0.58	−0.11 [-0.21, -0.01]	1.71 / 1.09 (25 / 75)
audio	-1.32	-0.45	+0.88 [+0.53, +1.26]	+0.01 [-0.17, +0.20]	-0.68	-0.92	−0.24 [-0.44, -0.05]	2.22 / 0.78 (15 / 20)
svg	-1.09	-0.54	+0.55 [+0.26, +0.84]	−0.06 [-0.30, +0.22]	-0.70	-0.87	−0.17 [-0.37, +0.05]	1.68 / 1.37 (5 / 55)
apiwf	-0.11	-0.29	−0.19 [-0.31, -0.07]	−0.19 [-0.30, -0.10]	-0.05	-0.25	−0.20 [-0.31, -0.10]	1.23 / 1.14 (5 / 0)
GPT-5.4	-0.95	-0.27	+0.68 [+0.38, +0.99]	+0.20 [-0.07, +0.46]	-0.52	-0.35	+0.17 [-0.07, +0.41]	1.67 / 0.80 (0 / 0)
Sonnet-5	-0.95	-0.30	+0.66 [+0.38, +0.95]	+0.01 [-0.23, +0.27]	-0.68	-0.65	+0.03 [-0.27, +0.37]	1.27 / 0.49 (0 / 0)
Haiku-4.5	-0.61	-0.75	−0.14 [-0.50, +0.22]	−0.58 [-0.89, -0.26]	-0.27	-0.64	−0.37 [-0.58, -0.15]	2.04 / 1.13 (5 / 0)
Gemini-3.5	-1.00	-0.65	+0.36 [-0.13, +0.78]	+0.04 [-0.56, +0.55]	-0.70	-0.77	−0.07 [-0.68, +0.38]	1.36 / 1.18 (0 / 10)
Qwen3.8-Max	-0.86	-0.34	+0.53 [+0.16, +0.90]	+0.01 [-0.75, +0.39]	-0.45	-0.86	−0.42 [-0.77, -0.05]	1.64 / 0.51 (0 / 0)
DeepSeek-V4-Pro	-0.94	-0.29	+0.65 [+0.31, +1.00]	−0.05 [-0.35, +0.24]	-0.42	-0.48	−0.05 [-0.25, +0.19]	2.36 / 2.40 (10 / 50)
Qwen3.5-27B	-1.03	-0.37	+0.66 [+0.23, +1.11]	+0.02 [-0.24, +0.27]	-0.44	-0.40	+0.04 [-0.16, +0.26]	2.13 / 0.93 (5 / 0)
Gemma-4-31B	-0.38	-0.46	−0.08 [-0.37, +0.23]	−0.28 [-0.49, -0.08]	-0.21	-0.58	−0.37 [-0.60, -0.13]	1.20 / 1.31 (5 / 15)

Table 21: Reasoning two-arm study: coverage slopes ($x = K/8$) by domain and model; *interaction* = slope with reasoning minus slope without (β_3; positive = flatter with reasoning). The All and domain rows pool eight vendor-specific reasoning configurations; the model rows and their intervals show how the interaction varies across them (on the count scale, five of the eight exclude zero). Count = terminal residual defects (the interaction regression); ANCOVA interaction = the $x \cdot \text{on}$ term after adding r_0 and $r_0 \cdot x$, parseable drafts; normalized residual = terminal residual divided by the trajectory's own round-0 defects r_0 (defects introduced during repair included; parseable drafts with $r_0 \geq 1$). Last column: mean round-0 defects per draft, reasoning off / on (an unparseable draft counts as eight), with the number of trajectories whose draft is unparseable in parentheses (five per such draft).

	normalized-residual interaction (Table 21)	survival of round-0 defects (parseable, $r_0 \geq 1$) off	on	interaction
All	-0.11 [-0.21, -0.01]	-0.51	-0.52	-0.01 [-0.10, +0.07]
audio	-0.24 [-0.44, -0.05]	-0.78	-0.86	-0.09 [-0.28, +0.11]
svg	-0.17 [-0.37, +0.05]	-0.73	-0.89	-0.17 [-0.33, +0.01]
apiwf	-0.20 [-0.31, -0.10]	-0.06	-0.16	-0.10 [-0.16, -0.03]
GPT-5.4	$+0.17$ [-0.07, +0.41]	-0.54	-0.36	$+0.18$ [-0.01, +0.37]
Sonnet-5	$+0.03$ [-0.27, +0.37]	-0.69	-0.67	$+0.02$ [-0.27, +0.36]
Haiku-4.5	-0.37 [-0.58, -0.15]	-0.33	-0.58	-0.25 [-0.42, -0.08]
Gemini-3.5	-0.07 [-0.68, +0.38]	-0.70	-0.50	$+0.20$ [+0.01, +0.39]
Qwen3.8-Max	-0.42 [-0.77, -0.05]	-0.45	-0.78	-0.33 [-0.67, +0.02]
DeepSeek-V4-Pro	-0.05 [-0.25, +0.19]	-0.43	-0.49	-0.06 [-0.24, +0.16]
Qwen3.5-27B	$+0.04$ [-0.16, +0.26]	-0.61	-0.39	$+0.22$ [+0.06, +0.41]
Gemma-4-31B	-0.37 [-0.60, -0.13]	-0.27	-0.56	-0.29 [-0.52, -0.06]

Table 22: Reasoning two-arm study, survival scale: the number of a draft's own defects still failing at the terminal state, divided by the draft's defect count r_0, regressed as in Table 21; the normalized-residual interaction of Table 21 is repeated for comparison. Pooled and domain rows use the joint domain×task cluster bootstrap; model rows resample tasks within the model.

Fixed-draft repair study. To separate the draft stage from the repair stage, both repair branches of a (model, task) start from the same reasoning-off draft (the reasoning-off arm's draft above) and switch reasoning only in the revision calls: the ON branch uses the model's reasoning-on policy and the OFF branch the reasoning-off control, under the same 32,768-token allowance, decoding settings, system prompt, and directives as the two-arm study. Coverage is $K \in \{2, 8\}$ (the task's main-grid K=2 mask; K=8 exposes all eight checks). Every branch runs exactly three revision rounds with no all-clear stop; each round shows the K-check report, even when every reported check passes, followed by the revision directive when a reported check fails and otherwise by the bare revision instruction of the fixed-horizon controls, *"Revise the artifact. Return the complete revised artifact in a single fenced code block."* The two branches of a (task, K) run back to back in a seeded random order: 8 models × 45 tasks × 2 K × 2 branches = 1,440 trajectories. Writing F_t for the set of checks failing at round t (so F_0 holds the draft's defects), the outcomes at round 3 (round 1 secondary) are the residual defect count $|F_3|$; the repair rate $|F_0 \setminus F_3|/|F_0|$, undefined when $|F_0| = 0$, with its exposed and hidden components; and defects introduced during repair, $|F_3 \setminus F_0|$. Estimands: the equal-weight mean of paired ON−OFF differences over (model, task) units at each K, their average over the two K, and the interaction $I = (\bar{y}_{\mathrm{ON},2} - \bar{y}_{\mathrm{ON},8}) - (\bar{y}_{\mathrm{OFF},2} - \bar{y}_{\mathrm{OFF},8})$, where $\bar{y}_{b,K}$ is branch b's mean round-3 residual count at K (negative = reasoning narrows the K=2 disadvantage; the two-arm interaction, defined on the slope in $K/8$, expresses the same direction with a positive sign). CIs: per domain, the 15 task clusters are resampled with replacement, each carrying all models, K values, and branches (B=10,000, seed 0); model rows resample tasks within the model. The three primary contrasts (round-3 residual ON−OFF, round-3 repair ON−OFF, and the residual interaction) are Holm-corrected, all at $p < 0.005$; domain, model, K, round-1, and sensitivity rows are secondary (Table 23). Unparseable revisions are scored under the rules of Appendix A (an unparseable artifact fails all eight checks): 8 ON and 5 OFF of the 720 trajectories per branch contain one (ON and OFF: Gemini-3.5 4 and 1, DeepSeek-V4-Pro 3 and 2, Haiku-4.5 1 and 1, Qwen3.8-Max 0 and 1); one Gemma-4-31B ON trajectory did not terminate within the run's time budget, so 179 of Gemma-4-31B's 180 trajectories completed and its pair is dropped from the paired estimates, which use Gemma-4-31B's remaining 89 pairs (178 trajectories). Dropping the pairs (the ON and OFF trajectories of one (model, task, K)), or the (model, task) quadruples (its four trajectories at K=2 and K=8), with an unparseable revision leaves the residual interaction at -0.25 (pairs) or moves it to -0.26 (quadruples), moves the K=8 residual difference from -0.38 to -0.36 (all CIs exclude zero), and moves no other estimate by more than 0.02. The residual gap between K=2 and K=8 is 0.56 [0.45, 0.68] without reasoning in repair and 0.31 [0.23, 0.40] with it. On the same drafts, reasoning in the repair stage raises repair of hidden defects, which exist only at K=2, by 44.4 pp [39.4, 49.9] and repair of exposed defects, averaged over K=2 and K=8 (where every defect is exposed), by 19.4 pp [15.8, 23.4] (20.3 pp at K=2 alone), leaves the number of defects introduced during repair unchanged overall (+0.01 [−0.02, +0.04] per trajectory), and, in the K=8 runs, repairs defects in the checks that the K=2 mask hides 17 pp [13, 20] more often than without reasoning.

residual defects, ON−OFF	K=2	K=8	overall	interaction I
All	−0.63 [-0.74,-0.53]	−0.38 [-0.48,-0.29]	−0.51 [-0.60,-0.42]	−0.25 [-0.34,-0.16]
audio	−1.17 [-1.38,-0.97]	−0.72 [-0.91,-0.55]	−0.94 [-1.12,-0.79]	−0.45 [-0.64,-0.26]
svg	−0.58 [-0.82,-0.36]	−0.28 [-0.45,-0.12]	−0.43 [-0.62,-0.25]	−0.31 [-0.45,-0.18]
apiwf	−0.15 [-0.25,-0.06]	−0.16 [-0.25,-0.03]	−0.15 [-0.23,-0.07]	+0.01 [-0.14,+0.13]
GPT-5.4	−0.56 [-0.73,-0.38]	−0.20 [-0.38,-0.04]	−0.38 [-0.52,-0.23]	−0.36 [-0.56,-0.16]
Sonnet-5	−0.07 [-0.24,+0.13]	−0.20 [-0.36,-0.07]	−0.13 [-0.26,-0.01]	+0.13 [-0.09,+0.38]
Haiku-4.5	−0.71 [-0.93,-0.49]	−0.47 [-0.67,-0.27]	−0.59 [-0.76,-0.44]	−0.24 [-0.51,+0.02]
Gemini-3.5	−0.51 [-0.69,-0.33]	−0.11 [-0.27,+0.04]	−0.31 [-0.44,-0.18]	−0.40 [-0.60,-0.18]
Qwen3.8-Max	−1.02 [-1.29,-0.76]	−0.67 [-0.87,-0.49]	−0.84 [-1.04,-0.66]	−0.36 [-0.62,-0.09]
DeepSeek-V4-Pro	−1.22 [-1.62,-0.87]	−1.00 [-1.49,-0.53]	−1.11 [-1.51,-0.77]	−0.22 [-0.69,+0.16]
Qwen3.5-27B	−0.56 [-0.76,-0.36]	−0.22 [-0.40,-0.07]	−0.39 [-0.53,-0.24]	−0.33 [-0.56,-0.11]
Gemma-4-31B	−0.41 [-0.60,-0.20]	−0.20 [-0.33,-0.09]	−0.30 [-0.43,-0.17]	−0.21 [-0.40,-0.00]
All, round 1	−0.54 [-0.68,-0.40]	−0.55 [-0.67,-0.44]	−0.55 [-0.66,-0.43]	+0.01 [-0.11,+0.14]
All, complete pairs	−0.62 [-0.71,-0.52]	−0.36 [-0.45,-0.28]	−0.49 [-0.57,-0.42]	−0.25 [-0.34,-0.16]
All, parseable quadruples	−0.62 [-0.72,-0.52]	−0.36 [-0.45,-0.28]	−0.49 [-0.57,-0.42]	−0.26 [-0.35,-0.16]

repair rate, ON−OFF (pp)	K=2	K=8	overall
All	+36.3 [+32.4,+40.0]	+18.6 [+15.6,+21.9]	+27.4 [+24.6,+30.3]
audio	+53.6 [+45.6,+60.9]	+22.9 [+18.6,+27.5]	+38.2 [+33.8,+42.4]
svg	+43.9 [+36.2,+50.8]	+14.2 [+6.8,+21.9]	+29.0 [+22.2,+35.5]
apiwf	+13.0 [+9.0,+17.3]	+18.3 [+13.8,+23.0]	+15.7 [+12.4,+19.3]
GPT-5.4	+34.7 [+25.6,+43.6]	+10.7 [+0.8,+21.0]	+22.7 [+15.1,+30.1]
Sonnet-5	+4.5 [-4.6,+13.9]	+14.0 [+4.2,+24.1]	+9.2 [+2.4,+16.0]
Haiku-4.5	+31.8 [+24.1,+39.6]	+14.4 [+7.1,+22.9]	+23.1 [+16.7,+30.3]
Gemini-3.5	+47.3 [+36.0,+58.1]	+8.5 [-1.2,+19.1]	+27.9 [+19.2,+36.4]
Qwen3.8-Max	+64.8 [+52.5,+76.8]	+39.8 [+28.8,+51.2]	+52.3 [+43.9,+60.9]
DeepSeek-V4-Pro	+48.2 [+39.0,+58.5]	+35.3 [+26.8,+43.9]	+41.8 [+35.4,+48.5]
Qwen3.5-27B	+20.5 [+13.1,+28.6]	+8.1 [+1.4,+15.3]	+14.3 [+8.6,+20.6]
Gemma-4-31B	+35.3 [+24.2,+45.7]	+18.6 [+5.6,+31.1]	+26.9 [+18.1,+35.8]
All, round 1	+34.6 [+31.3,+37.8]	+28.5 [+24.5,+32.5]	+31.6 [+28.8,+34.3]
All, complete pairs	+35.6 [+31.8,+39.2]	+18.6 [+15.7,+21.6]	+27.1 [+24.3,+29.8]
All, parseable quadruples	+35.9 [+32.1,+39.5]	+18.4 [+15.4,+21.5]	+27.1 [+24.4,+29.9]

Table 23: Fixed-draft repair study: reasoning on vs. off in the repair stage on the same reasoning-off draft, three revision rounds, round-3 outcomes unless stated. Residual = terminal residual defect count (negative = fewer with reasoning); repair rate = share of the draft's defects repaired, over units with at least one draft defect (325 at K=2, 326 at K=8); overall = equal-weight average of the two K; I = the residual interaction (negative = reasoning narrows the K=2 disadvantage). 95% CIs: domain×task cluster bootstrap for pooled and domain rows, task bootstrap within the model for model rows; the three primary contrasts in the All row are Holm-significant. The last two rows (*complete pairs*, *parseable quadruples*) drop the pairs, or the (model, task) quadruples, that contain an unparseable revision; 7 of the 8 truncated revisions in this study are unparseable and are dropped with them.

Self-selected checks. Tables 24 and 25 give the per-domain and per-cell (model×domain) results behind Sec. 7.2. Each of the 720 pairs (8 models × 3 domains × 30 tasks) revises one main-grid round-0 draft for exactly two revision rounds, and each round exposes two checks not exposed before: in the self-selected arm the model picks them through the selection call of Appendix A, and in the random arm a fixed random schedule, drawn once per task for all models, picks them. Each pick fills one *slot* (two per round, 2,880 per arm); the *defect-hit rate* is the share of picks whose check was failing when picked.

	audio	svg	apiwf
Δ defect-hit (self−random)	+.134 [.107,.161]*	−.044 [−.071,−.018]*	+.138 [.116,.159]*
defect-hit rates (self / random)	42.0% / 28.9%	18.0% / 22.4%	27.4% / 13.6%
Δ terminal residual	−.017 [−.154,.125]	+.004 [−.129,.125]	+.008 [−.046,.079]
fully correct (self / random)	19 / 14	56 / 62	15 / 14
repair of round-0 defects (self / random)	26.2% / 23.4%	32.8% / 35.8%	11.0% / 11.0%
exec.-dependent share (model picks)	65%	30%	54%
exec.-dependent share (random picks)	48%	54%	48%
exec.-dependent share (defects)	63%	86%	94%

Table 24: Self-selection study, per-domain outcomes (1,440 trajectories in 720 matched pairs; task-paired stratified bootstrap B=10k; * = CI excludes zero, uncorrected). Pooled Δ terminal residual (primary): -0.001 $[-0.068, +0.064]$; domain effects Holm-corrected over the three domains; the execution-dependent shares carry no CIs. Repair of round-0 defects: share of the failing checks of the main-grid round-0 draft that pass at the end, with the defects of all eight models in the domain pooled (no CI); computed before rounding, the self−random differences are $+2.7$, -3.1, and 0.0 pp (audio, svg, apiwf), behind the bound of about three percentage points in Sec. 7.2. Denominators: self-arm fallback slots (invalid selections replaced by the schedule) are excluded from the self-arm defect-hit rate and model-picks share; the random arm uses all slots. Counting fallback slots as the model's picks (intention to treat) gives an audio hit difference of $+0.127$ $[+0.099, +0.154]$ and leaves svg and apiwf unchanged (34 of 2,880 self-arm slots, all Haiku-4.5). Fully-correct: trajectories, of the 240 per arm and domain, whose terminal artifact passes all eight checks. Restricting to pairs whose draft had defects gives a pooled Δ of $+0.008$ $[-0.064, +0.079]$ and dropping pairs with a truncated call -0.036 $[-0.094, +0.019]$; of the 24 model×domain cells, 21 include zero, 1 favors self-selection, and 2 favor random. Execution-dependent share: fraction of self-selected picks placed on execution-dependent checks (row *model picks*), the same fraction for the random schedule's picks (row *random picks*), and fraction of round-0 defects lying in them (row *defects*); these are the checks that require rendering or running the artifact (Sec. 7.2), i.e., whose verdict cannot be read from the artifact's text (audio: tempo, no_clipping, bass_present, section_contrast; svg: containment, visibility, contrast, coverage; apiwf: completeness, side_effect_conservation, argument_fidelity, dependency_order).

Model	audio	svg	apiwf
GPT-5.4	−.033 [−.367,.467]	−.100 [−.333,.167]	−.067 [−.200,.067]
Sonnet-5	−.200 [−.400,−.033]	−.067 [−.367,.233]	+.033 [−.167,.233]
Haiku-4.5	−.267 [−.633,.100]	−.033 [−.300,.267]	+.067 [.000,.167]
Gemini-3.5	+.267 [.067,.467]	+.133 [−.067,.367]	−.067 [−.167,.000]
Qwen3.8-Max	+.200 [−.300,.833]	−.167 [−.467,.133]	−.033 [−.100,.000]
DeepSeek-V4-Pro	−.267 [−.800,.167]	+.367 [.133,.567]	+.200 [−.067,.633]
Qwen3.5-27B	+.300 [−.167,.900]	−.267 [−1.100,.367]	−.033 [−.133,.067]
Gemma-4-31B	−.133 [−.300,.033]	+.167 [−.067,.433]	−.033 [−.100,.000]

Table 25: Self-selection per-cell (model×domain) Δ terminal residual (self−random, mean over the cell's 30 pairs, with 95% task-bootstrap CIs; uncorrected). Three of 24 cells exclude zero, in opposite directions (Sonnet-5 audio favoring self-selection; Gemini-3.5 audio and DeepSeek-V4-Pro svg favoring random).

Where the residual defects sit. In audio and API workflows, self-selection raised the defect-hit rate without a detectable change in the residual defect count (Table 24); splitting the terminal residual by whether the arm ever exposed the check describes where those defects sit. Per task (self−random, same bootstrap as Table 24): defects in checks the arm exposed are higher by $+0.358$ $[+0.250, +0.467]$ in audio and $+0.529$ $[+0.454, +0.604]$ in apiwf, and defects in checks it never exposed lower by 0.375 $[0.263, 0.488]$ and 0.521 $[0.433, 0.608]$; the two components sum to the near-zero Δ terminal residual of Table 24. In svg, the one domain whose picks land on failing checks *less* often than the schedule, both signs reverse (-0.167 $[-0.263, -0.075]$ and $+0.171$ $[+0.050, +0.292]$). Of the round-0 defects an arm did report, the self arm repaired a smaller share in audio (32.5% vs. 38.7%, -6.1 pp $[-12.3, +0.7]$) and apiwf (11.3% vs. 17.4%, -6.1 pp

[−12.5, +0.7]), and a larger one in svg (67.4% vs. 59.5%, +8.0 pp [−0.8, +16.3]); these rates and all differences in this paragraph weight the eight models equally, as the intervals do. The split is descriptive and does not identify why the additional reports were not converted into repairs: a check enters the exposed group partly because the model picked it for being defective.

Probe diagnostics (selected subset). These diagnostics examine why API-workflow defects resist repair (Sec. 5.3). Subset: ten API-workflow tasks unsolved at $K=8$ by any of GPT-5.4, Sonnet-5, Haiku-4.5, Gemini-3.5, and Qwen3.8-Max; the three open-weight models did not solve them at $K=8$ either, so no model did. Probe experiment (Qwen3.8-Max at $K=8$, reasoning off, with the loop's all-clear stop and five-round horizon; the 10 tasks are run in two independent blocks, 20 runs per arm). In the *real-probe* arm the model may run up to two probe programs per run in a throwaway copy of the task's sandbox and receives the API calls they made with their actual responses; the *sham* arm receives the same offer and turn structure but fixed-size filler with no runtime values; the *no-probe* arm receives only the check report. Fully correct runs: 10/20, 3/20, and 1/20 (real probe: 5 of 10 in each block); an audit of every returned record found no content derived from the grading. Replay diagnostic (those five models, same ten tasks): each model's failing main-grid $K=8$ program is revised in a single call, once with reasoning off and once with it on, from a report naming the calls that a *known-good trace* (the API calls of a program that passes all eight checks) makes but the model's program never made (50 revisions per setting). With reasoning off, 70% of the named calls (1,071/1,536, over the 49 revisions that are not fully correct) were still never executed after the revision (an upstream divergence: the program leaves the known-good path before reaching them); with reasoning on, 45% (610/1,355, over 42 such revisions; three others were rejected before execution), fully correct revisions rise from 1/50 to 5/50, and counterproductive extra calls (calls made by neither the known-good trace nor the unrevised program) rise from 74 to 300. A documentation control (the main-grid protocol with the API manual extended, by script, with the return-value structure of all 453 APIs; Qwen3.8-Max on the ten tasks at $K=2$ and $K=8$, 20 runs) changes almost nothing relative to the main-grid runs of the same task and K (terminal residual defects: 18 ties, 1 better, 1 worse) and identified answer-only failures (Appendix B) in 8 of its 20 runs (world state fully correct; the only failure was a string in the answer field where the assertion suite expects null), so a correct world state and passing the whole assertion suite differ on this subset and part of what `completeness` counts as a defect is AppWorld's answer format rather than a wrong execution. These diagnostic results apply only to this subset and these models.

E TOKEN COSTS (DESCRIPTIVE)

This appendix gives the token costs behind the efficiency comparison in Sec. 5 and the report share cited in Sec. 4, counted in *model-side tokens*: the prompt and completion tokens returned by each model's API, summed over all calls of a trajectory. In the main grid (reasoning off), a $K=8$ trajectory costs on average 2.6× the model-side tokens of a $K=2$ trajectory (86.8k vs. 32.9k; longer reports and more rounds), but the fully correct rate rises from 11.8% to 47.7% (audio 11→59%, svg 19→74%, apiwf 5→10%), so tokens per fully correct artifact (all model-side tokens of the main-grid trajectories at that K, pooled over models and domains, divided by the number of those trajectories that end fully correct) fall by 35% overall (280k → 182k); computed per model, $K=8$ is cheaper for seven of eight models (by 10–63%) and 15% dearer for Gemma-4-31B. The saving comes from succeeding more often, not from cheaper repairs: among trajectories that ended fully correct, the median per-trajectory cost at $K=8$ was 1.4–4.5× that at $K=2$, depending on the model (most $K=2$ successes are drafts that were already correct), and on the few tasks whose defective draft a model repaired to fully correct at both $K=2$ and $K=8$ (3–10 per model), the median paired difference in repair cost ($K=8$ minus $K=2$) was under 0.5k tokens for six of eight models (GPT-5.4 +7.7k, Haiku-4.5 −5.4k). In the reasoning-on arm of the reasoning two-arm study (Sec. 7.1, Appendix D), $K=8$ uses 12–51% fewer reasoning tokens (as returned by each API) per fully correct artifact than $K=2$ for five models (Haiku-4.5 51%, Gemini-3.5 17%, Qwen3.8-Max 12%, Qwen3.5-27B 51%, Gemma-4-31B 22%) and 1% and 7% more for GPT-5.4 and DeepSeek-V4-Pro (Sonnet-5's API returns zero reasoning tokens on every $K=8$ call, so no ratio is given); counting all model-side tokens instead, $K=8$ is dearer per fully correct artifact for three models (GPT-5.4, DeepSeek-V4-Pro, Qwen3.5-27B). In the main grid, the average prompt grows from 10.5k tokens at revision round 1 to 25.3k at round 5, and the check report's share of its prompt, averaged over all K, models, and domains, falls from 3.2% at round 1 to 1.2% at round 5; the 36% of trajectories that never reach

an all-clear stop (they end at the horizon with an exposed check still failing) absorb 67% of all tokens. Among rounds 1–4 at which some exposed check still fails, the share of these rounds whose trajectory later reaches an all-clear stop (not necessarily a correct one) falls with the number of immediately preceding rounds in which the exposed checks' results did not change (0/1/2/$\geq$3 rounds): 40.6/30.6/17.5/4.5%.

F WORKED EXAMPLES: PROMPTS AND CHECK REPORTS

This section shows, verbatim, what a model sees in one task per domain, and then one real $K{=}2$ trajectory per domain from the main grid that ends in a false completion (the audio one is the example of Figure 1; only the API-workflow trajectory is on the task whose prompt is shown). Report lines (headed `INSTRUMENT REPORT` in the prompt) are shown without the trailing spaces that pad each line to a fixed width (Appendix A); lines longer than the page are wrapped. On a failing line, the text after the vertical bar is the check's counterexample.

Audio: the full initial prompt for task a3_000. The task was derived from the PDMX musical score *MiniBoss Theme 1* (two parts, 41 seconds): its tempo, key, and length are copied into the specification, and six thresholds are derived from the score (94 is the score's own count of out-of-scale notes, 40 semitones is the score's melody span plus four, and the stepwise and density thresholds are the score's values with a margin; this task's contrast threshold takes the fixed floor of 1.05, because 0.9 times the score's own ratio of 1.07 falls below it). The model never sees the source score itself. The system prompt is the one quoted in Appendix A; the user message is:

```
Write a program in the audio DSL that satisfies EVERY requirement below.

You compose music as a JSON program for a deterministic synthesizer.
Schema:
{"tempo_bpm": <int>, "meter": [4,4],
 "tracks": [{"name": "<str>", "timbre": "sine|square|saw|tri",
             "notes": [[midi_pitch, start_beat, dur_beats, velocity], ...]}],
 "sections": [{"name": "A", "start_beat": 0, "end_beat": <b>},
              {"name": "B", "start_beat": <b>, "end_beat": <end>}]}
Notes: midi_pitch 21-108, velocity 1-127, beats are floats. The synthesizer
sums tracks WITHOUT normalisation, so total amplitude is your responsibility.
Reply with ONLY the JSON object, no prose, no code fences.

Compose a piece of exactly 24 bars in 4/4.
- Tempo: 140 BPM.
- Key: C# minor; at most 94 notes outside the scale.
- Include a bass line (energy below 200 Hz must be clearly present).
- The audio must not clip.
- Two sections A then B of equal length; B must be audibly more intense
  (RMS at least 1.05x of A).
- The melody track must span at most 40 semitones.
- At least 11% of consecutive melody intervals must be <=2 semitones.
- Overall at least 4.04 notes per beat across all tracks.
- Tempo is accepted within 4% of the target.
- "Bass clearly present" is measured as a low-band (<200 Hz) energy ratio
  strictly above 0.0246.

Return ONLY the JSON object.
```

SVG: the requirements for task s2_015. The task is built from an SVGenius graphic with 19 elements. The construction damaged six checks (accessibility, attributes, containment, coverage, ids_preserved, visibility) and left structure and contrast intact. The user message is the svg initial prompt quoted in Appendix A, followed by the requirements below and the damaged document (7.9k characters, omitted here).

```
- structure: group 's2grp' must exist with exactly these direct children (and no
  transform on the group): s2e5, s2e6
- attributes: element 's2e1' must have fill="#4e7ab6" exactly
- ids_preserved: every one of these element ids must be present exactly once: s2e0,
  s2e1, s2e10, s2e11, s2e12, s2e13, s2e14, s2e15, s2e16, s2e17, s2e18, s2e2, s2e3,
  s2e4, s2e5, s2e6, s2e7, s2e8, s2e9, s2grp
- accessibility: the <svg> must begin with <title>artifact 108</title> followed by a
  non-empty <desc>
- containment: element 's2e4' must render entirely inside the viewport with >= 171px
  margin (512x512 render)
- visibility: at least 91% of element 's2e12' must be visible (not occluded) in the
```

```
  final composition
- contrast: composited colors of 's2e10' vs 's2e11' must have WCAG contrast ratio
  >= 3.93
- coverage: element 's2e0' must occupy between 16.42% and 18.22% of the canvas area
```

API workflows: the full initial prompt for task fd1f8fa_5. The task is an instance of AppWorld task family fd1f8fa ("remove the liked, or not liked, songs from the queue, then play"), with the parameter set to "liked". The system prompt is the one quoted in Appendix A; the user message is below, with the API manual (489 lines, 52k characters), which is identical for every task, elided.

```
You are operating a personal digital assistant for a supervisor. Complete the
task below by writing Python code that calls the app APIs listed in the manual.

TASK: Remove all the songs that I have liked from my Spotify queue, and then
start the player.

Current date and time: 2023-05-18T12:00:00

Rules:
- Call APIs as `apis.<app>.<api>(...)`; the return value is the response. `apis`
  is predefined in the runtime - use it directly, never `import apis`.
- Log in to each app you use (except supervisor/api_docs) and pass its
  access_token to every call of that app.
- Passwords come from apis.supervisor.show_account_passwords(); the username is
  the supervisor's email (phone app: phone_number).
- Use only the APIs in the manual; do not import modules, read files, or access
  anything outside `apis`.
- When fully done, call apis.supervisor.complete_task().

API MANUAL:
# AppWorld API MANUAL (identical for every task; * = required)
# Conventions: every non-supervisor app call needs access_token from
# app.login(username, password); passwords via
# supervisor.show_account_passwords(); usernames are the supervisor's
# email (phone app uses phone_number).
[... 489 lines elided ...]

Reply with ONE fenced Python code block containing the complete program
(```python ... ```). No text outside the fence is executed.
```

Three real $K{=}2$ trajectories that end in false completion. In each case the model repairs the one failure it is shown, the two exposed checks then pass, the loop stops, and at least one hidden check is still failing. The report shown is the one sent after the round-0 draft; the hidden verdicts are what the harness scored at the same time but did not report.

Audio, DeepSeek-V4-Pro, task a3_019, exposed checks bass_present and section_contrast.

```
INSTRUMENT REPORT (revision round 0)
- bass_present       PASS | requirement satisfied
- section_contrast   FAIL | section contrast: RMS(B)/RMS(A) = 1.01, required >= 1.05

Revise the artifact to fix the reported failures. Return the complete revised
artifact in a single fenced code block.
```

Hidden at round 0: density FAIL (2.25 notes/beat, required $\geq$ 2.42), melody_range FAIL (span 19 semitones exceeds 18), no_clipping FAIL (peak amplitude $\geq$ 1.0 in the raw mix); the other three hidden checks pass. After one revision both exposed checks pass and the loop stops. Hidden at that terminal round: density now passes (repaired without being reported), melody_range still fails (span 21 semitones exceeds 18), no_clipping still fails. Terminal state: two residual defects, a false completion.

SVG, DeepSeek-V4-Pro, task s2_032, exposed checks attributes and contrast.

```
INSTRUMENT REPORT (revision round 0)
- attributes         PASS | requirement satisfied
- contrast           FAIL | contrast: ratio 1.57 < 4.5

Revise the artifact to fix the reported failures. Return the complete revised
artifact in a single fenced code block.
```

Hidden at round 0: coverage FAIL (element s2e5 covers 10.98% of the canvas, required 29.6–31.4%); the other five hidden checks pass. After one revision the contrast ratio meets the threshold,

both exposed checks pass, and the loop stops. The coverage measurement is unchanged at the terminal round. Terminal state: one residual defect, a false completion.

API workflows, GPT-5.4, task fd1f8fa_5 (the prompt above), exposed checks argument_fidelity and precondition_reads.

```
INSTRUMENT REPORT (revision round 0)
- argument_fidelity  FAIL | 422: {"message":"The sort_by parameter must start with + or -."}
- precondition_reads PASS | requirement satisfied

Revise the artifact to fix the reported failures. Return the complete revised
artifact in a single fenced code block.
```

Hidden at round 0: completeness FAIL (AppWorld's own task-outcome assertions for the task, on the answer and on the MusicPlayer state, fail); the other five hidden checks pass. After one revision the rejected `sort_by` argument is fixed, both exposed checks pass, and the loop stops. The completeness assertions still fail at the terminal round. Terminal state: one residual defect, a false completion.

G LIMITATIONS

Output truncation affected 1.6% of main-grid trajectories; excluding truncated records makes one of the 15 GS-over-$K{=}2$ gains of Sec. 6.3 non-significant and leaves the reasoning-study and self-selection conclusions unchanged. The self-selection study uses two rounds at one coverage level (4/8), no early stopping, and one-line picks of check names; and the matched continuation control tests one message pair, always running the GS branch first (Appendix C). We study the verification-and-revision step of self-improvement, not a full pipeline. Throughout, the model cannot run any check itself, so our results describe correction loops in which the model has no independent access to the omitted verification signals; agents that can execute tests, render outputs, probe the environment, or write their own verifiers may recover more hidden defects, as the probe experiment on ten API-workflow tasks in Appendix D suggests.