AGENTS VERIFY THEIR ASSUMPTIONS, NOT THE SPECIFICATION: INTERPRETATION LOCK-IN IN AUTONOMOUS CODING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have become capable coding agents that complete long software tasks autonomously and test their own work. While LLMs excel at well-defined programming tasks such as code generation and bug fixing, their self-verification breaks down when a requirement is ambiguous, because the tests they write share the interpretation of the code they test. In this paper, we study this failure, which we call *interpretation lock-in*: agents commit to one reading of an ambiguous requirement, implement it, and validate their work against that same reading. In a controlled suite where every valid reading is known, and in pipelines rebuilt from long-horizon benchmark runs, the resulting programs crash or silently produce wrong output under other valid readings. The failure is largely one of commitment rather than knowledge: asked only to list plausible readings, the same models often name the one they missed. To remedy this failure, we propose reading-diverse self-training (RDST), which rewards an agent's own solutions for the number of valid readings they handle and fine-tunes the agent on its best episodes. Experimental results show that, relative to rewarding success on a single reading, RDST substantially reduces lock-in, including on task families held out from training, at little cost on well-specified tasks, and that the behaviour carries over to the interaction protocol of another agent harness.

1 INTRODUCTION

Large language models (LLMs) are now part of everyday software development, helping developers complete functions, fix bugs, and navigate large codebases. Recently, LLM agents have achieved great success in autonomous software engineering, resolving real repository issues and completing long-horizon terminal tasks with little human involvement (Jimenez et al., 2024; Yang et al., 2024; Merrill et al., 2026; Li et al., 2026), and they are increasingly trained on agent trajectories (Yang et al., 2025; Pan et al., 2025; Yang et al., 2026). Working alone, such an agent writes its own tests, runs them, and decides on that basis when the work is done. While these agents perform well when a task is fully specified, real specifications are rarely complete, and the resulting programs can pass the agent's own tests yet fail on inputs the specification allowed. Consequently, how agents verify their own work under incomplete specifications remains an underexplored area.

In this paper, we focus on ambiguous requirements in coding tasks, which refer to statements in a task specification that announce that inputs may vary without saying how. Typical examples include "timestamps may be given in other common formats" and "the manifest may arrive wrapped or encoded". The author of such a statement knows what they meant, but the text admits several concrete readings, and a program that is correct for the example input can still be wrong for the inputs the statement was warning about. Unlike an interactive assistant, an autonomous agent in the middle of a long task usually has no one to ask for clarification (Mu et al., 2024; Vijayvargiya et al., 2026a;b), and code models tend to settle such gaps by committing to a single interpretation (Richter & Papadakis, 2026). For such tasks, a natural question arises: when an agent tests its own work, does it test the requirement, or only its own reading of it?

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

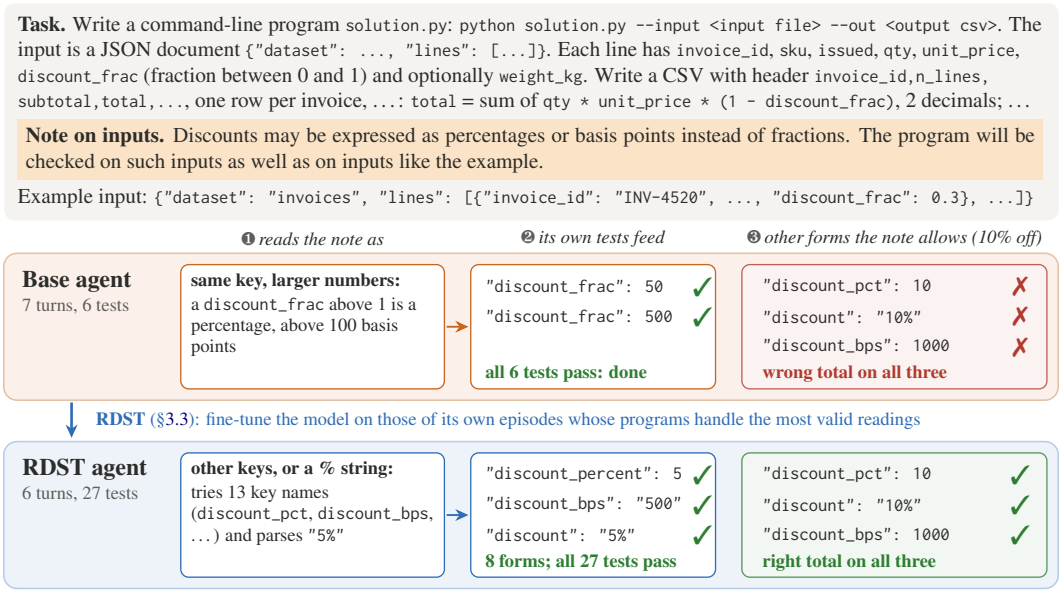


Figure 1: **Interpretation lock-in, and what RDST changes.** Two real episodes of Qwen3.8-27B on one task (abridged), before and after RDST; the task family was not used in training, though other training families announce percentages. The base agent commits to one reading of the note, tests only that reading and stops when its tests pass; its program gets the same 10% discount wrong in three other allowed forms. After RDST, the agent tests eight forms and gets all three right.

We find that agents almost never test a reading other than their own. The tests are written by the same reasoning that chose the reading, so they inherit it; the agent's tests pass, it stops, and nothing in its own feedback signals that the reading was one of several (Figure 1). We call this *interpretation lock-in*: the agent commits to one reading, implements it, and validates against that same reading. More tests of the chosen reading do not help, because the missing reading is exactly what those tests never exercise. Lock-in appears in every model we study, both in a controlled suite and on two long-horizon benchmarks, Long-Horizon-Terminal-Bench and Terminal-Bench 2.0. It is mostly a failure of commitment, not knowledge: when asked to enumerate plausible readings, the agent often lists the one its program missed.

Because the gap is mostly one of commitment, we close it in training rather than with a prompt. We propose *reading-diverse self-training* (RDST). On tasks whose specifications are automatically made ambiguous, every valid reading is known, so each of the agent's own episodes can be scored by how many readings its final program handles, a metamorphic oracle that needs no labels (Chen et al., 1998). RDST keeps each task's best episode, relabels it in hindsight so that the agent learns to behave this way without being prompted (Andrychowicz et al., 2017; Zhang et al., 2023), and fine-tunes the agent on the result.

Our main contributions are threefold:

- We build a controlled suite in which every valid reading of an ambiguous requirement is known, and find that agents rarely test readings they did not implement (1% of such readings for Qwen3.8-27B), in three models and on two long-horizon benchmarks.
- We show that lock-in is mostly a failure of commitment rather than knowledge: asked only to list plausible readings, Qwen3.8-27B names 58% of the readings its programs miss.
- We propose RDST, which needs no stronger model, labels or prompt at inference and raises the share of readings Qwen3.8-27B handles from 0.59 to 0.76, twice the gain of selecting the same episodes by success, including on task families never seen in training.

2 Related Work

Ambiguity in code generation. Beyond the semantic collapse of code models on underspecified prompts (Richter & Papadakis, 2026), work on ambiguity has focused on clarification, which can be elicited or trained (Mu et al., 2024; Vijayvargiya et al., 2026a;b), while agents that do not ask guess or commit silently (Stoisser et al., 2026; Ji et al., 2026); independent agent implementations even share failures traced to ambiguous specifications (Ron et al., 2026). These works measure whether the agent seeks clarification; we study the setting in which no one can be asked, where the committed reading also shapes the agent's own tests.

Self-verification. Models struggle to correct themselves without external feedback (Huang et al., 2024); generated tests help select code (Chen et al., 2023), but agent-written tests change the process more than the outcome (Chen et al., 2026), patches that pass them often fail hidden ones (Ahmed et al., 2026), and agents given the checks build to them rather than to the request (Ma et al., 2026); concurrent work locates silent failures of long-horizon agents with a separate verifier model (Rahman et al., 2026). These studies show that self-generated checks are weak evidence of correctness; we identify one reason under ambiguity, that the tests inherit the interpretation, and measure it with a metamorphic relation (Chen et al., 1998) over readings. Related evaluations apply semantics-preserving transformations to the task (Honarvar et al., 2026; de Zarzà et al., 2026); we apply the relation to the agent's own program, as measurement and as training signal.

Training coding agents. Software agents are fine-tuned on agent trajectories that resolve the task (Pan et al., 2025; Yang et al., 2025), i.e. rejection-sampling fine-tuning (Zelikman et al., 2022; Gulcehre et al., 2023) with success as the filter, or trained with reinforcement learning on terminal tasks (Yang et al., 2026). Our success-selected baseline is this recipe; RDST changes the filter to the number of valid readings handled and adds hindsight relabelling (Andrychowicz et al., 2017; Zhang et al., 2023).

3 Methodology

3.1 Problem Setting

A task x consists of a specification, an example input x_0 and its expected output y. An *announcing sentence* in the specification states that inputs may vary along some axis without fixing how ("Timestamps may be given in other common formats"). A *reading* r is one concrete form the variation can take (epoch seconds, epoch milliseconds, ...). Let $R(x)$ be the set of valid readings and x_r the example's content rendered under reading r. When readings preserve content, every correct program p satisfies the metamorphic relation $p(x_r) = y$ for all $r \in R(x)$ (Chen et al., 1998), so correctness under every reading can be checked without further labels. A program p *handles* r if $p(x_r) = y$; $H(p) \subseteq R(x)$ is the set of readings it handles. An agent episode τ ends with a program p_τ. Along the way the agent's tests and ad-hoc runs feed inputs to the program; $V(\tau) \subseteq R(x)$ is the set of readings among those inputs, the readings the agent *verified*. We measure

- *readings handled*, $h(\tau) = |H(p_\tau)|/|R(x)|$ if $p_\tau(x_0) = y$ and 0 otherwise;
- the *verification asymmetry*, $P\big(r \in V(\tau) \mid r \in H(p_\tau)\big)$ against $P\big(r \in V(\tau) \mid r \notin H(p_\tau)\big)$, over the readings of episodes whose plain case is correct;
- the *undetected-failure rate*: the share of episodes that pass their own tests but miss a valid reading.

Interpretation lock-in is the state in which verification follows implementation: $P(r \in V \mid r \notin H)$ is near zero, so passing tests say little about the readings the program does not handle.

3.2 A Controlled Suite with Known Readings

Tasks. Twelve families of small data-processing programs (sensor summaries, trip legs, invoices, payroll, ...): read a JSON document, write a per-group CSV. Each task shows the specification, the example input and its expected output.

Ambiguity injection. One announcing sentence is added to the specification; the variant form is never shown. Seven kinds of variation have 3–4 readings each, 25 in total: units, wrapper, missing-

value markers, key case, dates, number format and percentages (Appendix D). All readings preserve content. The generator yields 78 task types, and fresh instances are drawn for each evaluation run. Four of the twelve families are never used for training, and the kinds form two folds, (a) units, wrapper and missing values and (b) the other four.

Harness. Programs run in a network-less container on the plain input and on each reading; outputs are compared with per-column numeric tolerance. A logging shim records every input file the agent's tests or ad-hoc commands feed to the program, so $V(\tau)$ ignores the tests' assertions.

Settings. *Single-turn*: program and tests in one reply. *Multi-turn*: a sandbox, one command per reply, at most 12 replies; the agent decides when to stop.

3.3 Reading-Diverse Self-Training (RDST)

Our proposed RDST turns the metamorphic relation into a training signal that needs neither a stronger model nor labels. It fine-tunes the agent on its own episodes in four steps: episode sampling, selection by readings handled, hindsight relabelling and fine-tuning. It is thus rejection-sampling fine-tuning (Zelikman et al., 2022; Gulcehre et al., 2023) whose filter is the metamorphic oracle.

Episode sampling. We sample multi-turn episodes of the policy itself on ambiguity-injected tasks from the training families. Because the recipe can only select what the policy produces, three quarters of the episodes carry the enumeration hint of §3.4 in the prompt as an exploration aid: hinted episodes handle more readings.

Selection by readings handled. We score each final program on every reading, which gives $H(p_\tau)$, and read $V(\tau)$ from the shim. Among the episodes $E^+(x)$ of task x that end with the agent's own tests passing, we keep

$$\tau^*(x) = \underset{\tau \in E^+(x)}{\arg\max} \ \big(|H(p_\tau)|,\ |V(\tau)| \big) \qquad \text{(lexicographic)}, \tag{1}$$

the episode whose program handles the most readings, ties broken by how many readings its tests exercise. The *success-selected* baseline instead keeps a uniformly random episode of $E^+(x)$ whose program is correct on the plain input x_0. This is the usual filter when software agents are fine-tuned on trajectories that resolve their task (Pan et al., 2025; Yang et al., 2025), and it checks only the example's reading. Both arms draw on the same pool, tasks and number of examples (one episode per task for Qwen3.8-27B, two for Qwen3.5-9B).

Hindsight relabelling. Following Andrychowicz et al. (2017) and Zhang et al. (2023), the final files of the selected episode become the target for the bare task prompt, without the hint, written in both the single-turn and the multi-turn format.

Fine-tuning. We fine-tune with LoRA (Hu et al., 2022), rank 16.

3.4 Inference-Time Baselines and the Enumeration Probe

Enumeration hint. A task-agnostic paragraph asks the agent to list the plausible concrete readings of an announced variation, handle all of them and test each (Appendix F).

Enumerate-then-implement. The model first lists 4–8 plausible concrete forms of the announced variation; its own list is then placed in the coding prompt with an instruction to handle every item and feed one test input per item.

Reasoning budget. The same agent with thinking enabled.

Enumeration probe. To separate awareness from commitment, the model is asked only to list the plausible forms, and a judge call (the same model at temperature 0, shown a before/after record pair) decides whether the list covers each hidden reading; on Qwen3.8-27B the judge agrees with a manual audit in 20 of 22 cases.

3.5 Measuring Lock-in on Real Benchmarks

Real benchmarks do not come with known readings, but many grade the deliverable with itemised hidden checks. There the unit is one hidden check within one trial: it is *correct* if the check

Table 1: Unprompted agents in the suite. Single-turn: 234 tasks × 4 samples; "verified" = fed to the program by the tests. Multi-turn: Qwen, 208 tasks, 416 episodes; gpt-oss, 234 tasks, 468 episodes.

	plain case correct	readings handled	every reading	P(verif. | handled) / P(verif. | not handled)	undetected failures
Qwen3.8-27B, single-turn	0.888	0.491	0.160	0.418 / 0.055	—
Qwen3.8-27B, multi-turn	0.981	0.583	0.216	0.425 / **0.010**	0.782
Qwen3.5-9B, single-turn	0.832	0.188	0.013	0.365 / 0.066	—
Qwen3.5-9B, multi-turn	0.957	0.238	0.005	0.125 / 0.021	0.996
gpt-oss-20b, single-turn	0.910	0.455	0.108	0.442 / 0.024	—
gpt-oss-20b, multi-turn	0.923	0.458	0.100	0.270 / **0.012**	0.877

passes and *verified* if any of the agent's own commands exercised the behaviour it tests. An LLM judge from another model family (gpt-oss-20b, temperature 0, fixed prompts) decides verification from the task instruction, the agent's commands with output excerpts, and the check's code or expected values; it never sees whether the check passed. We report the same asymmetry as in §3.1, $P(\text{verified} \mid \text{correct}) - P(\text{verified} \mid \text{not correct})$; a second judge (Qwen3.8-27B) re-judges a fixed 20% sample (Appendix H). Since the judge counts a check as verified when the agent tested the same feature under a different reading, this is a conservative proxy for lock-in.

4 Experiments

4.1 Experimental Setup

Models. We train Qwen3.8-27B and Qwen3.5-9B, and use gpt-oss-20b (OpenAI, 2025), from a second model family, for the phenomenon, the mechanism and the hint. All models sample at temperature 1.0 (top-p 0.95) with thinking off unless stated (details in Appendix I).

Benchmarks. Beyond the controlled suite, we evaluate on two long-horizon terminal benchmarks: Long-Horizon-Terminal-Bench (LHTB; Li et al., 2026), graded with dense rewards, on the 42 tasks compatible with our harness, and Terminal-Bench 2.0 (TB2; Merrill et al., 2026), 89 tasks graded by hidden tests. All benchmark runs use Qwen3.8-27B.

Statistics. Single agent runs can differ by several points (Bjarnason et al., 2026), so differences are paired over tasks, and we call a difference significant only when a paired bootstrap over tasks excludes zero at the 5% level (Appendix I). Contrasts between trained arms were named before the confirming seeds were run, after an early contrast failed to replicate (Appendix E).

4.2 Interpretation Lock-in in Unprompted Agents

Readings handled. Unprompted programs handle some readings of a variation and almost never others (Qwen3.8-27B, single-turn): space-separated date strings are handled by 0.96 of programs against 0.18 for epoch seconds and 0.01 for epoch milliseconds; decimal-point strings 0.90 against 0.01 for decimal commas; a whole-file gzip+base64 wrapper 0.67 against 0.03 for a payload field.

Verification asymmetry (Table 1). Multi-turn, Qwen3.8-27B verifies 43% of the readings its program handles and 1% of those it does not; 78% of the episodes whose own tests pass fail at least one valid reading (Qwen3.5-9B: 99.6%). The asymmetry replicates in the second family: gpt-oss-20b verifies 27% of the readings it handles and 1% of those it does not (single-turn: 44% against 2%). It often stops without running its tests at all (16% of its episodes end with them passing), and 88% of the episodes that do end with passing tests fail a valid reading. Execution feedback does not help by itself: the agent runs its program on the readings it implemented.

Long-horizon benchmarks (Figure 2). With hidden checks as the unit (§3.5), the unprompted agent's own commands exercised 39% of the checks it passed and 25% of those it failed on LHTB (1,089 checks on the 27 tasks whose verifiers itemise them; difference $+0.140$, analysis named in advance), and 50% against 11% on TB2 (260 checks on 74 tasks; difference $+0.386$). The second judge disagrees with the first on individual items (Cohen's κ 0.22 and 0.49 on the two benchmarks) and on how often agents verify at all, but finds the asymmetry on the same items (base agent: $+0.194$ against $+0.129$, and $+0.645$ against $+0.500$). They also disagree on why checks fail (κ 0.05; of

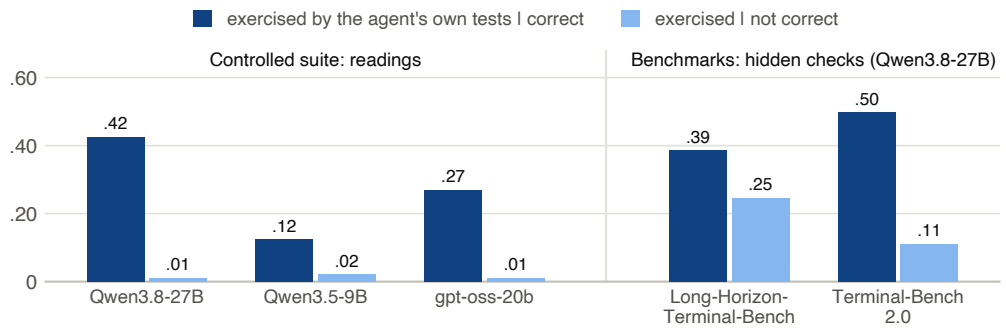


Figure 2: The verification asymmetry in every setting: the share of correct behaviour that the agent's own tests or commands exercised (dark), against the share of incorrect behaviour (light). Suite: valid readings the program handles or not, from logged test inputs of unprompted multi-turn agents. Benchmarks: hidden checks the unprompted Qwen3.8-27B agent passed or failed, as judged by an LLM that does not see the outcome.

nine failures that replays show to be interpretation failures, the first judge identified none as such), so we report no failure-type shares. The asymmetry is consistent with lock-in but does not by itself exclude the simpler account that agents fix what they test; either way, hidden failures concentrate in behaviour the agent never exercised. It is not only a difference between tasks: within tasks and within trials (post hoc) it is $+0.123$ and $+0.115$ on LHTB. TB2 tasks mostly pass or fail as a whole, so only 15 of them have both passed and failed checks, and the within-task estimate there is not informative ($+0.154$, not significant).

Rebuilt benchmark pipelines. On the two benchmark tasks whose instructions announce hidden variations in prose, 81 pipelines rebuilt from archived Qwen3.8-27B runs and run on perturbed public inputs fail along encoding lines; generic perturbations never cause failures (Appendix A): 37/39 materials pipelines crash on a Celsius boundary record, and 37/42 document pipelines decode the wrapper under the whole-file reading against 9/42 under the payload reading. The agents had tested the reading they implemented (29/50 materials runs wrote Celsius tests, one for a boundary record). Where an instruction *enumerates* the readings, all eight pipelines handle every listed unit.

4.3 Awareness versus Commitment

With the same task text, Qwen3.8-27B is asked only to *list* 4–8 plausible forms of the announced variation (§3.4). The list names the hidden reading 62% of the time (3,358 list–reading pairs), and 58% of the time for the readings that unprompted programs miss. Table 7 in Appendix C shows three regimes. *Named but not handled*: named in 50–94% of lists, handled in at most 17% of unprompted programs—a commitment gap. *Neither named nor handled*: millisecond epochs and decimal commas are rarely named and rarely handled—an awareness gap not resolved by the prompting approaches tested here. *Handled but not named*: generic parsers accept compact dates and dash or `NA` missing markers that lists seldom name. With its own list in the prompt and an instruction to support every item, the model handles 79% of the readings it named and 51% of those it did not, so an implementation gap remains. The same decomposition holds on the benchmark tasks (Appendix B).

Other models (Table 6 in Appendix C). In all three models a reading named in the model's own list is handled far more often than one it leaves out, so the commitment effect is not specific to one model. How much of the lock-in is awareness differs: gpt-oss-20b and Qwen3.5-9B name only 35% and 39% of the readings their unprompted programs miss, against 58% for Qwen3.8-27B. Qwen3.5-9B also fails many readings it does name, and the long own-list prompt costs it plain-case accuracy (0.83 to 0.64). Each model judges its own lists and only Qwen3.8-27B's verdicts were audited, so the other models' named rates are approximate.

Table 2: Qwen3.8-27B: base, success-selected and RDST (reading-selected) adapters from the same pool, trained on all seven kinds. Multi-turn, 234 tasks.

	plain case	readings handled	every reading	verified readings	P(verif. | handled) / P(verif. | not h.)	undetected failures	median replies
base	0.972	0.588	0.224	0.253	0.411 / 0.025	0.772	10
+ success-selected	0.998	0.669	0.303	0.351	0.495 / 0.058	0.695	4
+ RDST	0.989	**0.758**	0.368	0.489	0.593 / **0.162**	0.619	5

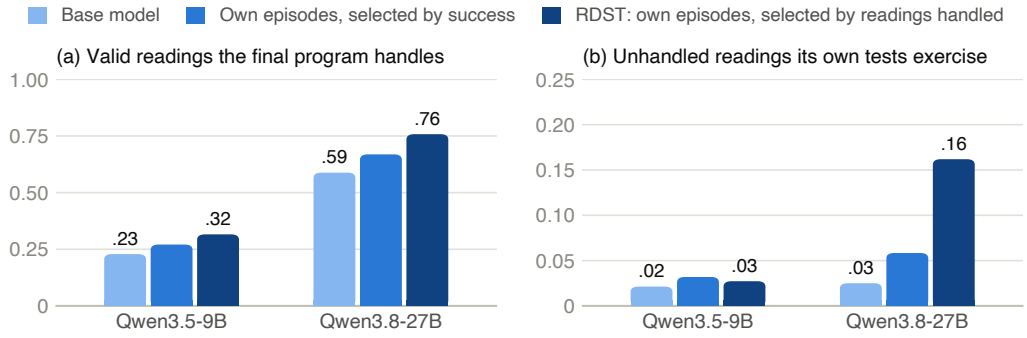


Figure 3: Training on the agent's own episodes (multi-turn). (a) Share of valid readings the final program handles. (b) Share of the readings the program does *not* handle that the agent's own tests or ad-hoc runs exercise, P(verified | not handled). Values in Tables 2 and 9.

4.4 Main Results

Gains over success-selection. On Qwen3.8-27B (Table 2; its own 1,664 multi-turn episodes, about seven candidates per task), RDST improves readings handled by $+0.170$ over base (named in advance) and by $+0.190$ on never-trained families, with no plain-case cost. The success-selected arm gives $+0.081$, so the selection signal adds $+0.089$ (never-trained families $+0.108$; verified readings $+0.138$): about half of the gain.

Transfer to held-out kinds. Trained on fold a only, Qwen3.8-27B also improves on the four kinds it never saw: RDST against base $+0.046$ on keys, dates, number format and percent (named in advance), with verified readings $+0.098$. The success-selected arm transfers as much ($+0.047$; difference -0.001, not significant): at this scale, training on the agent's own hint-explored episodes carries over to new kinds of ambiguity whichever selection is used, while reading-selection still adds on the trained kinds ($+0.043$) and on never-trained families ($+0.072$).

Results at a smaller scale. On Qwen3.5-9B (trained on fold a; Table 9 in Appendix E; 1,664 own episodes; selected-episode coverage 0.52 against 0.34 under success-selection), RDST improves on base by $+0.087$ and by $+0.136$ on never-trained families. Against success-selected it adds $+0.045$, again about half of the gain (trained kinds $+0.094$; never-trained families $+0.087$; the other fold's kinds $+0.007$, not significant). Verified readings rise too ($+0.033$ against success-selected, $+0.037$ on untrained kinds). No plain-case cost, no turn inflation.

Changes in agent behaviour. With RDST, Qwen3.8-27B's verification asymmetry itself shrinks: the agent checks 16% of the readings it has *not* handled, against 3% for the base model (Figure 3b); Qwen3.5-9B's does not change. Both Qwen3.8-27B adapters finish in 4–5 replies instead of 10: hindsight targets teach it to write the program and its tests at once. Per reading (Appendix D), the gains come from readings the hinted policy sometimes produced (epoch seconds, percent keys, unit fields); readings it almost never produced (decimal commas, unit strings) stay near zero.

4.5 Comparison with Inference-Time Baselines

The hint raises readings handled by $+0.083$ single-turn and readings fed to the tests by $+0.120$; enumerate-then-implement handles as many readings and tests more of them (Table 10 in Appendix E). Multi-turn the hint gives $+0.093$, about half of what training gives the same model,

Table 3: Ablations of the recipe, evaluated multi-turn (‡ single-turn; ^{j} in the JSON protocol of the benchmark harness). Coverage: share of readings handled by the selected training episodes, success-selected / reading-selected (RDST). Differences in readings handled, paired over tasks; plain: RDST minus base, plain case. †: not significant (§4.1).

variant	coverage	reading − base	reading − success	plain
Qwen3.8-27B, main recipe, all kinds	0.68 / 0.86	+0.170	+0.089	+0.017
Qwen3.5-9B, main recipe, fold a	0.34 / 0.52	+0.087	+0.045	+0.015
Qwen3.5-9B, trained on fold b	0.34 / 0.38	+0.033	+0.026	+0.003
Qwen3.5-9B, hinted single-shot pool‡	0.49 / 0.95	+0.101	+0.019†	−0.043
Qwen3.5-9B, stronger model's samples	0.78 / 0.96	+0.145	+0.028†	−0.064
Qwen3.5-9B, second round	0.40 / 0.61	+0.111	+0.049	−0.023
Qwen3.8-27B, two-turn JSON targets^{j}	—	+0.019†	—	−0.006
Qwen3.8-27B, full JSON trajectories^{j}	0.66 / 0.76	+0.108	+0.043	+0.009
Qwen3.8-27B, fold a, held-out kinds only	0.72 / 0.88	+0.046	−0.001†	+0.015

and $P(\text{verified} \mid \text{not handled})$ rises from 0.010 to 0.112. A reasoning budget raises readings handled from 0.57 to 0.73 (40 tasks) but leaves the asymmetry in place (0.50 against 0.07). For Qwen3.5-9B and gpt-oss-20b the hint helps on units, wrappers and missing values and not on the other kinds (Appendix E): how much a prompt helps depends on the model and on the kind of ambiguity.

4.6 Ablation Studies

Table 3 varies one part of the recipe at a time.

Selection filter. The filter matters when success-selected data do not already cover many readings: across rows, the gain over base tracks how far the coverage of the selected data lies above the base policy's. In agentic pools, randomly chosen successful episodes cover little more than the unhinted policy does (Qwen3.5-9B: 0.34), and selecting by readings handled is what changes the model. In a pool of hinted single-shot samples, success-selection already delivers high-coverage data (0.49); it amounts to context distillation of the hint (Snell et al., 2022), and selecting by readings adds only $+0.019$ (not significant). On fold b there is little to select (0.38 against 0.34), and the gain is small.

Distillation from a stronger model. A stronger model's samples also reduce lock-in, but reading-selection is not better than success-selection there, and imitating the teacher's long programs costs Qwen3.5-9B 18–33 points of plain-case accuracy (single-turn; Appendix E).

Iteration. A second round improves both arms and the gap persists, but fewer episodes finish.

Target format. Two-turn hindsight targets in the benchmark harness's JSON protocol give a null adapter. Full multi-turn trajectories in that protocol, trained in an independent run (Appendix E), reduce lock-in there ($+0.108$) and again favour reading- over success-selection ($+0.043$), though this adapter stays below RDST (-0.035), whose protocol transfer replicates in that run's evaluation ($+0.143$, against $+0.142$ in §4.7).

Held-out kinds. Training on the agent's own episodes transfers to kinds never trained on at both scales (RDST against base: $+0.046$ on Qwen3.8-27B, $+0.031$ on Qwen3.5-9B), but on neither model does selection by readings add to it (-0.001 and $+0.007$ against success-selection).

4.7 Transfer to Long-Horizon Benchmarks

Interaction protocol. On the suite run in the JSON command protocol of the benchmark harness, RDST gives $+0.142$ over base (0.571→0.713; contrast named in advance), $+0.205$ on never-trained families and $+0.178$ in verified readings, with no plain-case cost: the behaviour carries over to an interaction protocol it was not trained in.

Benchmark reward. We compare RDST with base first on a 15-task panel of LHTB fixed before this study, then on the benchmark's 27 other compatible tasks, registered separately before any of their trials. The enumeration hint (appended to each instruction) is compared with base on the panel. The analysis (per-task means, an exact paired sign-flip test over tasks, Holm across the two comparisons) was fixed in writing before any run was complete (Appendix G). On the 15-task panel

Table 4: Long-horizon benchmarks: LHTB under the clean protocol and TB2 (the 65 tasks scored in both arms). Difference: mean over tasks of the per-task difference in mean reward; p: exact paired sign-flip test over tasks. Panel: 15 tasks fixed before this study; 27 others: the rest of the 42 compatible tasks. *Registered as primary. One cell of the 27-task run is missing (its audit timed out).

vs. base	base	treated	difference	p	tasks + / −
RDST, panel	0.346	0.369	$+0.023$	0.23	7 / 6
hint, panel	0.352	0.371	$+0.020$	0.53	7 / 7
RDST, 27 others*	0.217	0.211	-0.006	0.70	10 / 14
RDST, all 42	0.263	0.268	$+0.004$	0.70	17 / 20
RDST, TB2	0.431	0.446	$+0.015$	$\approx$1.0	6 / 5

neither comparison is significant (Table 4): RDST gives $+0.023$ ($p = 0.23$) and the hint $+0.020$ ($p = 0.53$). RDST's estimate fell as runs accumulated ($+0.195$ after the first, $+0.066$ with $p = 0.02$ at the interim look, $+0.023$ at the end): several tasks land either near 0 or near 0.8–0.9 per trial, so single-run comparisons on them are unreliable (Bjarnason et al., 2026). On the 27 other tasks it gives -0.006 ($p = 0.70$), and over all 42 tasks $+0.004$: the behaviour trained on the suite does not raise reward on this benchmark. The same holds on TB2 ($+0.015$ over the 65 tasks scored in both arms, $p \approx 1$). Nor does it shift verification on LHTB: its own commands exercised 33% of the hidden checks it passed and 20% of those it failed ($+0.128$; base 39% and 25%).

The two tasks with documented lock-in. No arm changes rewards or handled readings; unshown encodings, unnamed readings and hard algorithmic requirements gate reward (Appendix B).

5 Conclusion

We propose RDST, a self-training recipe that rewards an agent's own solutions for the number of valid readings of an ambiguous requirement they handle and fine-tunes the agent on its best episodes, relabelled in hindsight. It reduces interpretation lock-in, which we find in every model we study and on two long-horizon benchmarks: agents commit to one reading, implement it and verify only that, so their passing tests say little about whether the requirement was read correctly. RDST transfers to unseen task families and another interaction protocol, needs no stronger model, labels or prompt at inference, and on Qwen3.8-27B helps about twice as much as prompting, without reducing plain-case accuracy in the main experiments. Lock-in is mostly a gap of commitment, which a training signal over readings can close.

AI Use Statement

We used LLM-based tools to help with coding, drafting and proofreading the paper, to assist in designing and analysing the experiments and to help build the synthetic task suite.

Reproducibility Statement

Section 3 and Appendices D–I specify the task generator, readings, harness, metrics, training recipe and benchmark protocol. Every number in the paper is produced by scripts that read the stored episodes; the suite generator, harness, training and analysis code will be released.

References

Toufique Ahmed, Jatin Ganhotra, Avraham Shinnar, and Martin Hirzel. Investigating test overfitting on SWE-bench. In *Companion Proceedings of the ACM International Conference on the Foundations of Software Engineering*, 2026.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, 2017.

Bjarni Haukur Bjarnason, André Silva, and Martin Monperrus. On randomness in agentic evals. *arXiv preprint arXiv:2602.07150*, 2026.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. CodeT: Code generation with generated tests. In *International Conference on Learning Representations*, 2023.

Tsong Yueh Chen, Shing Chi Cheung, and Siu Ming Yiu. Metamorphic testing: A new approach for generating next test cases. Technical Report HKUST-CS98-01, Hong Kong University of Science and Technology, 1998.

Zhi Chen, Zhensu Sun, Yuling Shi, Chao Peng, Xiaodong Gu, David Lo, and Lingxiao Jiang. Rethinking the value of agent-generated tests for LLM-based software engineering agents. *arXiv preprint arXiv:2602.07900*, 2026.

I. de Zarzà, J. de Curtò, Jordi Cabot, Pietro Manzoni, and Carlos T. Calafate. Semantic invariance in agentic AI. *arXiv preprint arXiv:2603.13173*, 2026.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (ReST) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

Shahin Honarvar, Amber Gorzynski, James Lee-Jones, Harry Coppock, Marek Rei, Joseph Ryan, and Alastair F. Donaldson. Capture the flags: Family-based evaluation of agentic LLMs via semantics-preserving transformations. *arXiv preprint arXiv:2602.05523*, 2026.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

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*, 2024.

Zimo Ji, Zekai Zhang, Congying Xu, Yujia Tian, Zongjie Li, Yudong Gao, Shuai Wang, and Shing-Chi Cheung. Coding agents are guessing: Measuring action-boundary violations in underspecified DevOps instructions. *arXiv preprint arXiv:2607.02294*, 2026.

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

Zongxia Li, Zhongzhi Li, Yucheng Shi, Ruhan Wang, Junyao Yang, Zhichao Liu, Xiyang Wu, Anhao Li, Yue Yu, Ninghao Liu, Lichao Sun, Haotao Mi, and Leowei Liang. Long-Horizon-Terminal-Bench: Testing the limits of agents on long-horizon terminal tasks with dense reward-based grading. *arXiv preprint arXiv:2607.08964*, 2026.

Yanuo Ma, Ben Kereopa-Yorke, and Ben Schultz. Building to the test: Coding agents deliver what you check, not what you requested. *arXiv preprint arXiv:2606.28430*, 2026.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, 2026.

Fangwen Mu, Lin Shi, Song Wang, Zhuohao Yu, Binquan Zhang, Chenxue Wang, Shichao Liu, and Qing Wang. ClarifyGPT: A framework for enhancing LLM-based code generation via requirements clarification. *Proceedings of the ACM on Software Engineering*, 1(FSE):2332–2354, 2024.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In *International Conference on Machine Learning*, 2025.

Salman Rahman, Yubin Kim, Mihir Parmar, A. Ali Heydari, Genglin Liu, Simon A. Lee, Weizhi Zhang, Arian Hosseini, Ahmed A. Metwally, Yuzhe Yang, Baharan Mirzasoleiman, Xin Liu, Pavel Izmailov, Saadia Gabriel, Mark Malhotra, Shwetak Patel, Daniel McDuff, and Hamid Palangi. Locating hidden failures makes long-horizon agents more reliable. *arXiv preprint arXiv:2609.17930*, 2026.

Cedric Richter and Mike Papadakis. Underspecification does not imply incoherence: The risks of semantic collapse in coding models. *arXiv preprint arXiv:2607.01953*, 2026.

Javier Ron, Benoit Baudry, and Martin Monperrus. N-version programming with coding agents. *arXiv preprint arXiv:2606.20158*, 2026.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

Josefa Lia Stoisser, Marc Boubnovski Martell, Sidsel Boldsen, Kaspar Märtens, and Robert Kitchen. Ambig-DS: A benchmark for task-framing ambiguity in data-science agents. *arXiv preprint arXiv:2605.09698*, 2026.

Sanidhya Vijayvargiya, Vijay Viswanathan, and Graham Neubig. Asking what matters: Reward-driven clarification for software engineering tasks. *arXiv preprint arXiv:2604.14624*, 2026a.

Sanidhya Vijayvargiya, Xuhui Zhou, Akhila Yerukola, Maarten Sap, and Graham Neubig. Ambig-SWE: Interactive agents to overcome underspecificity in software engineering. In *International Conference on Learning Representations*, 2026b.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems*, 2025.

Junyao Yang, Yucheng Shi, Zhongzhi Li, Ruhan Wang, Zongxia Li, Haitao Mi, and Leowei Liang. T1: Terminal agent reinforcement learning for long-horizon tasks. *arXiv preprint arXiv:2609.11042*, 2026.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

Tianjun Zhang, Fangchen Liu, Justin Wong, Pieter Abbeel, and Joseph E. Gonzalez. The wisdom of hindsight makes language models better instruction followers. In *International Conference on Machine Learning*, 2023.

Appendix Overview

Appendices A and B are case studies on Long-Horizon-Terminal-Bench, Appendices C–E give further analysis and figures, and Appendices F–I give implementation and evaluation details.

A Case Study: Rebuilt Long-Horizon Pipelines

Corpus. 1,811 scored trials of Qwen3.8-27B-FP8 on Long-Horizon-Terminal-Bench (Li et al., 2026) under the Terminus-2 agent (Merrill et al., 2026) and many controller variants of it. We study the two tasks whose scorecards split into public and hidden check groups and whose images are available: `materials-phase-diagram-audit` (50 trajectories; hidden groups carry

Table 5: Pipelines rebuilt from archived long-horizon runs, under perturbations of the public inputs. "Silent" = exit 0 with empty or wrong output.

task	perturbation	fail	mode
materials (39)	relocate / reverse key order (generic)	0/39	—
	Celsius on *invariant* records	13/39	12 crash
	Celsius on *boundary* records	37/39	34 crash
	Celsius as value + unit field	39/39	36 crash
	gzip+base64 payload (encoding not public)	38/39	silent empty
document (42)	relocate / key order / decoy OCR	0/42	—
	wrapper: whole-file base64+gzip / raw gzip	5/42, 9/42	crash
	wrapper: JSON object with `payload`	33/42	silent empty
	nested-asset / token schemas (not public)	40–41/42	mostly silent

≈0.93 of the reward) and `document-table-layout-reconstruction` (46; 0.98). Both instructions announce the hidden variations in prose—composition units, Celsius records, nested or external manifest payloads, gzip+base64 wrappers, alternate schemas—without fixing their encoding.

Replay. For each trial we start the pinned task image, replay every keystroke block of the archived trajectory in a persistent shell, and run the deployed entrypoint on the public inputs. Replays reproduce the archived public outputs byte for byte for 39/46 materials and 42/44 document trials; the rest are excluded. Each perturbation transforms the *public* inputs, and a metamorphic relation states what the output must do (invariance, non-emptiness, exit status). One early job had verifier logs readable mid-run; its 10 affected trials are excluded.

Inferable and grader-specific encodings. We separate encodings *inferable* from the public material (Celsius on any record; both wrapper readings) from grader-specific ones (the materials payload; the nested schemas), and report both in Table 5; the lock-in signature is clearest on the inferable ones, which the agents had the material to anticipate.

Control task. A climate-audit task *enumerates* its readings instead of announcing variation: it lists three temperature units, five precipitation-unit spellings and two calendars, and its manifests declare units in fields. On the eight most recent complete archived base trials, a replay with perturbations that convert the public values and change the unit and calendar fields (relation: identical output) finds that every pipeline handles Celsius, Fahrenheit, the three alternative precipitation units and the no-leap calendar (8/8 each); the 360-day calendar was not tested. Its hidden groups nevertheless score 0.00, which this replay does not explain; the control therefore bears on which readings are handled, not on reward.

B Case Study: Interventions on the Two Lock-in Tasks

On the two tasks of Appendix A we ran four arms under the clean protocol of Appendix G: base, the enumeration paragraph appended to the instruction, RDST, and both. Rewards do not move (materials 0.06–0.09, document 0.02–0.06, in every arm), and rebuilding the 24 final pipelines with the replay instrument shows why: all handle exactly the same readings (document: whole-file gzip 12/12, payload wrapper 0/12, nested asset schema 0/12; materials: Celsius on invariant records 12/12, on boundary records 0/12, value+unit field 0/12).

Materials. None of the interventions we evaluated raises its reward: the whole hidden reward is gated by a gzip+base64 payload wrapper that the public material never shows (every arm's pipeline emits nothing for the hidden manifest), and the boundary-record Celsius reading the grader also uses is one the model never names when asked to enumerate on the task text and public manifest (0/8 lists; 6/8 name Celsius keys on other record types).

Document. This task shows the commitment gap on a real benchmark: the payload wrapper is named in 5/8 lists, yet 0/12 working agents implement it; every arm builds gzip/base64 test inputs under the whole-file reading. The appended paragraph never produced a list of readings in any trajectory. Putting the model's own list into the instruction made one pipeline handle the payload wrapper—the first in any of our runs—while two lost the whole-file reading that every other arm

handles: the checklist moved the commitment rather than widening it. The wrapper group carries only 0.02 of the reward; the rest also needs rotated or skewed pages and nested schemas. The interaction protocol does not explain this: in the benchmark harness's JSON protocol RDST keeps its effect on the suite (§4.7).

C The Enumeration Probe in Three Models

Table 6: The enumeration probe in three models (single-turn, about 3,350 list–reading pairs each). Own list: the model's own list is placed in the coding prompt with an instruction to handle every item.

	Qwen3.8-27B	gpt-oss-20b	Qwen3.5-9B
list names the hidden reading	0.623	0.471	0.436
among readings unprompted programs miss	0.580	0.352	0.388
own list: $P(\text{handled} \mid \text{named})$	0.794	0.793	0.547
own list: $P(\text{handled} \mid \text{not named})$	0.512	0.351	0.322
readings handled: own list (unprompted)	0.582 (0.491)	0.523 (0.455)	0.261 (0.188)
plain case: own list (unprompted)	0.857 (0.888)	0.955 (0.910)	0.639 (0.832)

Table 7: Readings named by the enumeration probe and readings handled (Qwen3.8-27B, single-turn). Hint: the enumeration hint. Own list: the model's own list is placed in the coding prompt.

reading	named	handled, no hint	handled, hint	handled, own list
wrapper: gzip+base64 payload	0.94	0.03	0.09	0.18
units: separate unit field	0.91	0.06	0.43	0.33
units: suffixed key	0.76	0.17	0.51	0.59
percent: basis points	0.67	0.01	0.06	0.18
number format: decimal comma	0.25	0.01	0.07	0.16
dates: epoch milliseconds	0.08	0.01	0.08	0.05
dates: compact ISO	0.14	0.95	0.88	0.82

Figure 4 shows, for Qwen3.8-27B, how often each reading of Table 7 is named against how often programs handle it, and Figure 5 how naming a reading changes whether it is handled in the three models.

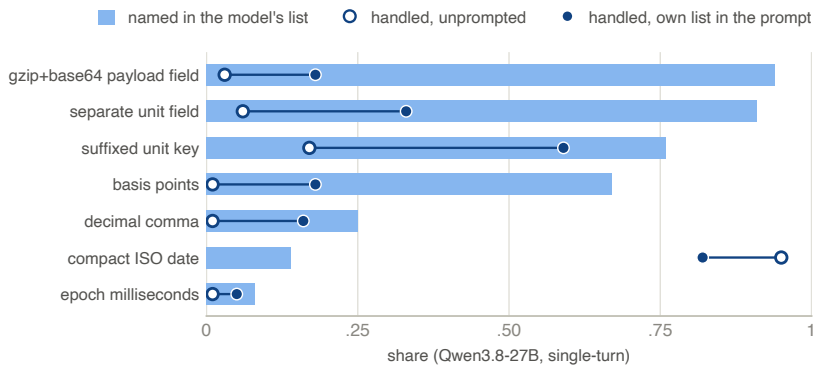


Figure 4: Readings named when asked, against readings handled (Qwen3.8-27B, single-turn; values of Table 7). Most of these readings are named in most lists yet handled by few unprompted programs; putting the model's own list in the prompt raises handling but leaves a large gap. Compact ISO dates are handled without being named, because generic date parsers accept them.

D Readings and Per-Reading Outcomes

Readings by kind: units (suffixed key in either alternative unit, separate unit field, value string with unit), wrapper (the document under a `data` or a `payload` key, a gzip+base64 payload field,

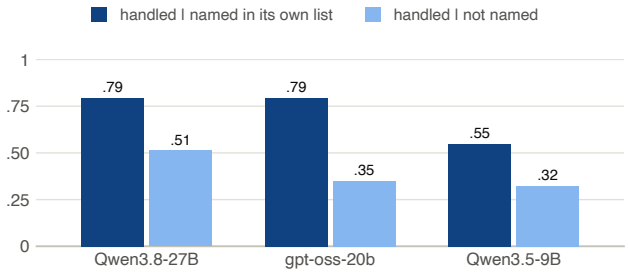


Figure 5: With its own list in the prompt, each model handles a reading it named far more often than one it did not (values of Table 6).

gzip+base64 whole file), missing values (`null`, empty string, `NA`, dash), keys (camel, upper, kebab), dates (epoch seconds, epoch milliseconds, space-separated, compact), number format (decimal string, decimal comma, scientific), percent (percent key, percent string, basis points). Table 8 gives the handled rate per reading for the runs of Tables 2 and 9. Readings with a zero base rate mostly stay near zero after training (millisecond epochs and decimal commas for Qwen3.5-9B, unit strings for both models): the recipe can only select what the policy already produces sometimes. On Qwen3.8-27B, RDST is the exception on epoch milliseconds ($0.01 \rightarrow 0.25$) and percent readings ($0.00 \rightarrow 0.42$–0.77), where the hinted policy produced candidates.

Table 8: Share of final programs handling each reading, multi-turn, unprompted, pooled over evaluation runs; same episodes as Tables 2 and 9. (g) marks the reading a fixed-convention grader would use.

	Qwen3.8-27B			Qwen3.5-9B		
reading	base	success	reading	base	success	reading
dates: compact	0.99	0.99	0.94	0.64	0.68	0.75
dates: epoch milliseconds	0.01	0.01	0.25	0.00	0.00	0.00
dates: epoch seconds (g)	0.36	0.50	0.78	0.00	0.00	0.00
dates: space-separated	1.00	1.00	1.00	0.83	0.86	0.83
keys: camel	0.93	1.00	0.94	0.11	0.10	0.10
keys: kebab	0.94	0.97	0.94	0.10	0.07	0.03
keys: upper (g)	0.47	0.92	0.83	0.00	0.02	0.02
missing: dash	0.83	0.88	0.97	0.01	0.10	0.23
missing: empty string	0.93	1.00	1.00	0.01	0.12	0.26
missing: `NA` (g)	0.85	0.90	0.99	0.01	0.10	0.23
missing: `null`	0.97	1.00	1.00	0.58	0.74	0.92
number format: decimal comma (g)	0.01	0.04	0.08	0.00	0.01	0.02
number format: decimal string	0.99	1.00	1.00	0.83	0.94	0.97
number format: scientific	0.99	0.94	0.88	0.83	0.93	0.97
percent: basis points	0.00	0.15	0.54	0.00	0.01	0.02
percent: percent key (g)	0.00	0.23	0.77	0.00	0.01	0.02
percent: percent string	0.00	0.06	0.42	0.00	0.00	0.00
units: value with unit string	0.03	0.02	0.03	0.00	0.01	0.00
units: suffixed key, first unit (g)	0.25	0.47	0.83	0.07	0.12	0.20
units: suffixed key, second unit	0.27	0.48	0.80	0.03	0.11	0.17
units: separate unit field	0.17	0.40	0.63	0.00	0.01	0.00
wrapper: under `data`	0.94	0.97	0.97	0.57	0.74	0.77
wrapper: under `payload`	0.96	0.97	0.97	0.40	0.48	0.52
wrapper: gzip+base64 payload field (g)	0.15	0.15	0.21	0.00	0.00	0.01
wrapper: whole-file gzip+base64	0.92	0.99	0.93	0.29	0.24	0.55

E BASELINES, DISTILLATION AND ITERATION

Table 9 gives the Qwen3.5-9B results of §4.4.

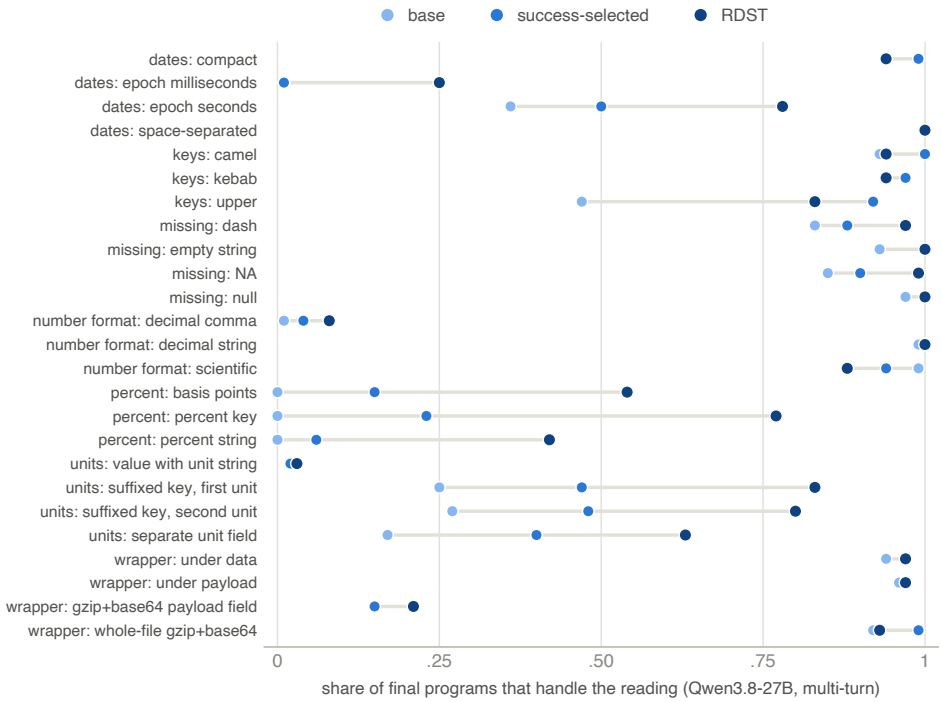


Figure 6: Share of final programs that handle each reading (Qwen3.8-27B, multi-turn, unprompted; the Qwen3.8-27B columns of Table 8). RDST gains most on readings the hinted policy sometimes produced (percent readings, suffixed unit keys, epoch seconds); readings it almost never produced, such as decimal commas and unit strings, stay near zero.

Table 9: Qwen3.5-9B trained on fold a (units, wrapper, missing values). Multi-turn, 234 tasks.

	plain case	readings handled	trained kinds	other-fold kinds	every reading	verified readings	$P(\text{verif.} \mid \text{handled})$ / $P(\text{verif.} \mid \text{not h.})$
base	0.951	0.229	0.174	0.271	0.002	0.043	0.128 / 0.021
success-selected	0.963	0.270	0.240	0.294	0.018	0.069	0.178 / 0.032
RDST	0.966	**0.316**	0.334	0.302	0.038	0.101	0.259 / 0.027

Harness. When gpt-oss emits a tool call that cannot be parsed, the harness resamples that reply; this never happened with the Qwen models.

Prompting baselines. Enumerate-then-implement against the hint: $+0.008$ readings handled (not significant), $+0.053$ readings fed to the tests. Multi-turn, the hint lowers the undetected-failure rate from 0.78 to 0.67 (Qwen3.8-27B). The reasoning budget was measured multi-turn on 80 episodes of an earlier 40-task set. Qwen3.5-9B, multi-turn: $+0.094$, of which $+0.189$ on units, wrappers and missing values and $+0.020$ on the other kinds. gpt-oss-20b: $+0.052$ single-turn ($+0.128$ on units, wrappers and missing values, -0.007 on the other kinds); multi-turn $+0.003$ (not significant), with $+0.078$ on the first three kinds and -0.055 on the other four.

Distilling a stronger model. Single-shot Qwen3.8-27B samples distilled into Qwen3.5-9B under four selection rules (task success; one fixed "grader" reading; hinted teacher; most readings handled). Every arm reduces lock-in on the trained kinds ($+0.12$ to $+0.22$ single-turn; $+0.12$/$+0.15$ multi-turn, including never-trained families), but the reading-diverse arm is not better than the others (single-turn it is worse than success-selection, -0.039), and imitating the teacher's long programs costs Qwen3.5-9B 18–33 points of plain-case accuracy single-turn (-0.04/-0.06, not significant, when the agent can run and repair its program). The gain is capability transfer.

On-policy single-shot samples. Qwen3.5-9B's own hinted samples, selected by success (coverage 0.49) or by readings handled (0.95). Success $+0.082$, reading-selected $+0.101$; never-trained families $+0.13$/$+0.15$; plain case -0.075/-0.043; reading-selected against success $+0.019$ (not

significant). On the first evaluation run alone this contrast was $+0.056$ and significant; it did not replicate, which is why all later contrasts were named in advance and confirmed on further runs. Here success-selection of hinted samples already delivers high-coverage data: it is context distillation of the hint.

Fold b (keys, dates, number format, percent), whose pool offers little to select (coverage of the selected episodes 0.38, against 0.34 under success-selection): reading-selected against base $+0.033$, of which $+0.057$ on the other fold's kinds.

Iteration. A second round (sampling from the round-1 policy on fresh instances, selecting from the union) lifts readings handled $0.236 \rightarrow 0.319 \rightarrow 0.347$ for the reading-selected arm and $0.236 \rightarrow 0.263 \rightarrow 0.298$ for the success-selected arm (Figure 7); the gap persists ($+0.049$), and both arms lose some plain-case accuracy (-0.02/-0.05) and finish fewer episodes.

Independent run in the JSON protocol. The base Qwen3.8-27B sampled 1,664 episodes with full transcripts in the benchmark harness's JSON protocol (training families, all seven kinds); per task the reading-selected or the success-selected episode was kept (coverage 0.76 and 0.66), one training example per agent reply with the hint removed from the first prompt, LoRA r=16, context up to 32k tokens. The adapters were evaluated in both protocols within that run; the primary contrast, named before the run, was RDST (JSON targets) against base in the JSON protocol.

Truncated episodes. When a model request fails (12–24% of the Qwen3.5-9B adapters' episodes, at most 5% of the base model's, none for the Qwen3.8-27B runs of Table 2), the episode ends and is scored on the program written so far; as this happens more often to the adapters, it tends to lower their scores relative to base. In the main tables at most one episode per arm never produced a scored program and is dropped.

Table 10: Inference-time baselines, Qwen3.8-27B, single-turn (234 tasks). "Tests feed" = share of readings the tests exercise.

condition	plain case	readings handled	every reading	tests feed
no hint	0.888	0.491	0.160	0.227
hint (list readings, handle all, test each)	0.876	0.574	0.253	0.347
enumerate-then-implement (own list)	0.857	0.582	0.281	0.400

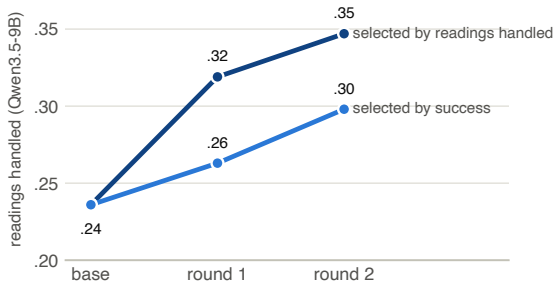


Figure 7: Iteration (Qwen3.5-9B): a second round of sampling and selection improves both selection rules, and the gap between them persists.

F Enumeration Hint

Suite: "Where the task says that inputs may vary without saying exactly how, do not settle on a single interpretation. First list the plausible concrete readings of that statement (for example different field names, encodings, value formats, or which records it applies to). Make the program handle all of them, and make the tests feed the program one input per reading." Benchmark: the same request appended to a copy of each instruction as a final section ("Handling ambiguity in this task description").

G BENCHMARK PROTOCOL

Harness tree frozen before this study (final verification only; interim hidden verification and privileged evaluator modes disabled); pinned task images; a pre-agent check that no grader file is visible in the container; a post-agent record of the verifier directory; base and treated arm launched simultaneously, three concurrent trials per arm; every response audited for references to the verifier directory. A first attempt that used the working copy of the harness, which feeds interim verifier results back to the agent and leaves the hidden scorecard readable, produced a base trial scoring 0.963 by reading it and was discarded. Pre-specified analysis (recorded before any seed was complete): per task and arm the mean reward over available seeds, missing rewards left missing; estimate = mean per-task difference; exact paired sign-flip test over tasks (two-sided); task-level bootstrap interval (10,000 resamples); secondary, the same over (task, seed) cells; Holm across the RDST and prompt comparisons (the smaller p must be below 0.025). The prompt comparison had an interim futility stop if its difference was not positive. Looks at the data before the final analysis are reported as interim.

H JUDGE PROTOCOL FOR THE BENCHMARK MEASUREMENTS

Items. One item is one hidden check within one trial: a pytest test function (parametrised cases aggregated; failed if any case fails or errors), or a scorecard group, gate or submitted stage, whichever the task's verifier exposes. Apex stages the agent never submitted are not items, and neither are scorecard groups whose checks never ran; both would count work the agent never produced as unverified. Such groups still count as failed checks for the trial-level metric. Tasks whose verifier reports a single score per trial (on the 15-task panel: generals, grammar-fuzz, snake, sudoku, tabular, unknown-config, vector-db) enter only the trial-level metric, *declared complete yet failing*: the agent's final response claims completion and the reward is below the task maximum.

Judge. For each item, gpt-oss-20b (a different model family from the agents, temperature 0, fixed prompts) is asked whether any of the agent's own commands checked the behaviour the hidden check tests. The judge sees the task instruction, the agent's commands with excerpts of their output, and the check's code or its expected values; it does not see whether the check passed, the trial's reward or the check's failure output. A separate call, made only for failed items, shows the failure output and asks why the check failed (interpretation, implementation, not attempted, other). The prompts were fixed before any item was judged and before any Terminal-Bench 2.0 data were collected.

Deviations. All deviations were logged before the verdicts were analysed. (i) Some gpt-oss answers at temperature 0 repeat one sentence until the token limit and carry no verdict; such answers, and only those, are asked again at temperature 0.7 (at most three times); 144 of 3,978 answers needed a retry (32 of 2,643 verification answers and 112 of 1,335 failure-type answers), and no item remains without a verdict. (ii) A first version of the item extractor counted scorecard groups whose checks never ran as passed items (85 on the 15-task panel); the definition above corrects it. All reported results use the corrected items; an interim summary of about 450 items computed before the correction is superseded. The correction was triggered by an extraction check on new trials, not by the verdicts.

Validation. A second judge, Qwen3.8-27B, re-judged a fixed 20% hash sample of items (436 on Long-Horizon-Terminal-Bench, 107 on Terminal-Bench 2.0). Raw agreement on whether a check was verified is 0.55 (κ 0.22) and 0.74 (κ 0.49); on why a failed check failed it is 0.55 and 0.40 (κ 0.05 on both); the second judge labels 29% of failures on Long-Horizon-Terminal-Bench as interpretation failures and the first 7%, figures that show how far the judges disagree, not how common interpretation failures are. On the same items both judges find the asymmetry (base agent, first and second judge: $+0.129$ and $+0.194$ on Long-Horizon-Terminal-Bench, $+0.500$ and $+0.645$ on Terminal-Bench 2.0). The verification verdicts have not been audited manually, so we use the judges only for the verification asymmetry, which both reproduce, and report no failure-type shares.

I Implementation and Evaluation Details

Implementation. Qwen3.8-27B is evaluated with FP8 weights, and its adapters are trained on the BF16 weights. Qwen3.5-9B is trained on fold a, Qwen3.8-27B on all seven kinds and, to test held-out kinds, on fold a alone.

Significance. Differences are paired over tasks. On the suite we call a difference significant when a paired bootstrap over tasks excludes zero at the 5% level; on the benchmarks we use the exact paired sign-flip test over tasks registered in Appendix G.