MC-RV CoT-Enhancer: Improving the Distillability of Reasoning Traces via Latent Step Risk Search

Anonymous ACL submission

Abstract

In Chain-of-Thought (CoT) distillation, strong teacher models transfer reasoning ability to smaller students through intermediate rationales, yet most pipelines assume that a verified-correct trace is automatically useful supervision. We show that this assumption already fails for trajectory prefixes: even when a full trajectory reaches the correct answer, some partial reasoning states along the way are fragile, in the sense that stochastic continuations from them often fail. We define this failure mode *Step-wise Fragility*, and argue that CoT distillability should be judged by how reliably a student can re-enact the reasoning process from partial prefixes. To optimize this situation, we propose *MC-RV CoT-Enhancer*, a search-and-validate framework that operates entirely within the verified-correct solution space. MC-RV decomposes a teacher trace into resumable checkpoints, uses a latent step-risk prior to locate likely bottlenecks, applies targeted local rewrites, and retains an edit only when Monte Carlo rollout validation improves *stochastic re-enactment success rate* (SRSR) while preserving final correctness and low deviation from the teacher trace. Rather than polishing traces globally or repairing wrong solutions, MC-RV stabilizes the specific transitions that make correct traces hard to learn. Experiments with a Qwen3-8B student establish new state-of-the-art results among reported sub-10B deep-research agents across six benchmarks. Process-level analysis further shows that MC-RV raises average SRSR from 55.7% to 64.9%, with larger gains on fragile steps, indicating that step-level stablization is a practical criterion for building more distillable reasoning supervision.

1 Introduction

Chain-of-Thought (CoT) traces have become a central interface for transferring reasoning ability from strong teacher models to smaller students (Wei et al., 2022; Zelikman et al., 2022; Hsieh et al., 2023). The usual assumption is simple: if a teacher trace reaches the correct answer and passes verification, then it is a suitable target for supervised distillation. This assumption is increasingly fragile in long-horizon settings (Sinha et al., 2025) . A student does not only imitate a finished solution; it must learn how to continue from many partial reasoning states. Therefore, a useful trace should be not only correct at the endpoint, but also replayable from its intermediate prefixes.

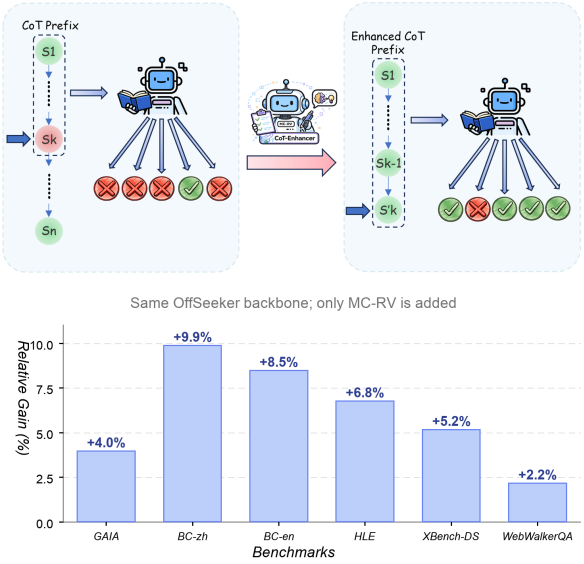

Figure 1: Correctness does not guarantee distillability: MC-RV CoT-Enhancer localizes latent step risks and rewrites them to improve replay robustness (as measured by stochastic re-enactment success rate) and downstream performance.

Recent work on CoT supervision has strengthened reasoning traces from trace construction to process verification. For trace construction, Chen et al. (2025b) introduce chain-of-step reasoning, decomposing coarse rationales into structured intermediate steps and using fine-grained rewards for step-level supervision and reinforcement learning. Niu et al. (2025) propose CoTD-PO, which lets the student explore its own trajectories and op-

timize toward preferred reasoning paths to reduce teacher-student distribution mismatch. Complementarily, process-supervision methods evaluate reasoning before the final answer: Lightman et al. (2024) train process reward models from human step-level labels, while Setlur et al. (2025) scale automated process verification with process advantage verifiers that reward steps by their progress toward a correct solution.

Despite this progress, deep-research distillation leaves two gaps that are not fully captured by final-answer correctness, trajectory preference, or local step verification. First, it lacks an explicit measure of *prefix-conditioned replay robustness*: whether a partial trajectory, or prefix, gives a student enough recoverable information to complete the remaining reasoning under stochastic decoding. This matters because deep-research traces often encode implicit decisions about query formulation, evidence selection, and stopping criteria. Second, even verified-correct traces can suffer *Step-wise Fragility*: the full trajectory reaches the right answer, but some intermediate prefixes remain weak supervision states because resampled continuations from them frequently derail the later search.

To address these limitations, we introduce *stochastic re-enactment success rate* (SRSR), a process-level metric that estimates whether a model can complete a verified-correct solution when conditioned on an intermediate reasoning prefix. Unlike final-answer accuracy, which only observes the endpoint, or local step correctness, which checks an isolated transition, SRSR measures whether a prefix supports reliable continuation under controlled stochastic rollouts. Built upon SRSR, we propose **MC-RV CoT-Enhancer**, a step-level stabilization framework for verified-correct CoT data. As shown in Figure 1, MC-RV first decomposes a teacher trace into resumable checkpoints, then uses a latent step-risk prior to identify likely bottlenecks, rewrites the selected local transition, and accepts the edit only when Monte Carlo rollout validation improves SRSR while preserving final correctness and low deviation from the teacher trace. This search-and-validate design keeps optimization within the correct-solution manifold while shifting the supervision target from endpoint correctness to replay-robust intermediate reasoning states. We evaluate MC-RV on long-horizon deep-research agents, where reliable continuation is essential for multi-step web search and evidence synthesis. With a Qwen3-8B student, MC-RV establishes new state-of-the-art results among reported sub-10B deep-research agents across GAIA, BrowseComp-zh/en, HLE, XBench-DS, and WebWalkerQA. Process-level analysis further shows that MC-RV improves average SRSR from 55.7% to 64.9%, with larger gains on fragile steps, supporting our central claim that improving the replay robustness of correct traces can translate into stronger student reasoning.

Our main contributions include:

- We identify *Step-wise Fragility*, where verified-correct CoT traces contain prefixes that are difficult for a student to continue reliably.
- We introduce SRSR as a process-level metric for prefix-conditioned replay robustness.
- We present **MC-RV CoT-Enhancer**, which stabilizes fragile checkpoints through latent step-risk search, targeted rewriting, and Monte Carlo rollout validation.
- We show that MC-RV achieves state-of-the-art reported sub-10B deep-research performance and improves replay robustness, especially on fragile steps.

2 Related Work

Chain-of-Thought (CoT) traces are typically improved through generation, certification, or post-hoc stabilization. These lines have made CoT data more accurate and easier to filter, but they mostly optimize final-answer correctness, step correctness, or whole-trajectory preference. Our work studies a complementary failure mode: *correct-but-brittle* traces whose intermediate prefixes are difficult for a student to continue reliably. This changes the unit of analysis from a completed rationale to a resumable reasoning state.

2.1 Generating Reasoning Traces

A first line of work improves reasoning supervision by strengthening how multi-step traces are *generated*. Prompting surveys cover elicitation and decomposition strategies for explicit intermediate reasoning (Schulhoff et al., 2024), while recent work casts generation as exploration over intermediate "thought" states or graph-structured artifacts (Sel et al., 2023; Besta et al., 2024). These methods broaden the search space of possible rationales and often recover solutions that direct decoding misses.

Consistency-based decoding further improves trace quality through aggregation or selection (Wang et al., 2023; Chen et al., 2024, 2023; Feng et al., 2025), and distillation-oriented work studies when generated rationales or difficulty-graded data help student models (Do et al., 2025; Tian et al., 2025). These methods improve solution discovery and dataset construction, but they usually score a trace after it is complete. MC-RV is different: it begins from verified-correct traces and asks whether their intermediate prefixes are stable continuation states for the student that will be trained on them.

2.2 Certifying Trace Correctness

A complementary line of work *certifies* generated traces for filtering, ranking, or search. Step-wise verification provides more diagnostic supervision than outcome filtering (Lightman et al., 2024), and verifiers or process reward models score intermediate steps for reranking, dense supervision, and first-error localization (Balcan et al., 2025; Xu et al., 2025; Yang et al., 2025). Such verifiers are valuable because they expose where a trajectory becomes invalid rather than only whether the final answer is wrong.

Self- or zero-shot verification reduces reliance on task-specific verifiers (Chowdhury and Caragea, 2025; Wang et al., 2024; Jia et al., 2025), and related schemes improve factual consistency or guide search (Dhuliawala et al., 2024; Yu et al., 2025). These methods primarily target outcome- or step-level correctness. MC-RV keeps correctness as a hard constraint, but uses stochastic re-enactment to test a second property: whether a certified-correct prefix is usable as supervision for continuation.

2.3 Auditing and Stabilizing Reasoning

Beyond certification, robustness and faithfulness audits examine whether reasoning remains stable under perturbations or faithful to the stated rationale (Zhou et al., 2024; Gupta et al., 2024; Paul et al., 2024). Stabilization methods refine outputs through verification loops, preference objectives, or targeted thinking-token interventions (Dhuliawala et al., 2024; Zhang et al., 2024; Wu et al., 2025b). This literature is closely related in spirit because it treats reasoning as a process rather than a single label.

Recent work also studies self-correction and intrinsic self-verification at training or test time (Ma et al., 2025; Lee et al., 2025; Wu et al., 2025b). These approaches improve outcome robustness or reduce factual and logical errors, but they typically operate at the whole-response or coarse trace level. Our work focuses on a finer intervention point: whether an intermediate prefix is *resumable* and *re-enactable*. MC-RV therefore localizes fragile checkpoints, edits only the risky transition, and validates the edit by randomized continuation rather than by a static score alone.

This also distinguishes MC-RV from preference optimization and RL-style process optimization. Those methods change the student policy through rewards, preferences, or test-time aggregation, whereas MC-RV changes the SFT trajectory distribution before training. Its ablations therefore compare original SFT data, generic rewriting, quality-only rewriting, and replay-validated rewriting under the same student and training budget, isolating the effect of making verified-correct traces more distillable.

3 Method

MC-RV CoT-Enhancer treats verified-correct traces as candidates for stabilization rather than automatically reliable supervision. Given an input x, a verified teacher trajectory y^T, and a target learner π_θ, MC-RV outputs either the original trajectory or a locally rewritten trajectory $y^\dagger$ whose prefixes are easier for the learner to resume. The method has four modules: decomposing the trajectory into step-localized units, prioritizing likely fragile checkpoints, rewriting the selected transition, and validating the edit with Monte Carlo re-enactment. Section 3.1 formalizes the objective through stochastic re-enactment success rate (SRSR), and Section 3.2 turns the objective into this bottleneck-focused search-and-validate procedure.

3.1 Theoretical Formulation

3.1.1 Correctness Is Not Distillability

Correctness is a necessary but insufficient condition for distillability. A teacher trajectory can pass final-answer and step-level verification while still containing prefixes from which a student cannot reliably resume under stochastic decoding. We call this gap **Step-wise Fragility** (SWF): a trajectory is correct as a complete solution, but brittle as prefix-conditioned supervision. In long-horizon research tasks, such brittleness often comes from

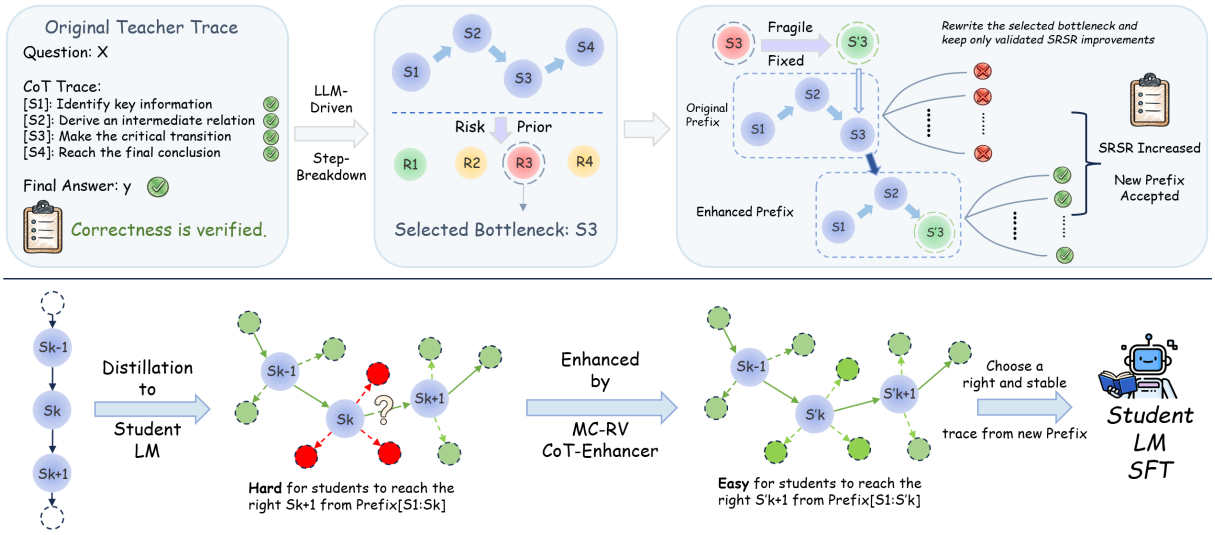

Figure 2: MC-RV CoT-Enhancer localizes fragile checkpoints in verified-correct teacher traces, rewrites the selected transition, and accepts the edit only when Monte Carlo rollout validation improves SRSR while preserving correctness and semantic proximity to the teacher trace.

compressed evidence transitions, underspecified search intents, or latent assumptions that are clear only after seeing the teacher's later steps.

3.1.2 A Distillability-Constrained Objective

Let $p_T(y \mid x)$ be the teacher-induced trajectory distribution and $q(y \mid x)$ the rewritten distribution used for student training. MC-RV keeps rewrites close to the teacher while favoring trajectories that are easier to continue:

$$\max_{q \in \mathcal{Q}} \; \mathbb{E}_{x \sim \mathcal{X}} \mathbb{E}_{y \sim q(\cdot \mid x)} [\mathrm{SRSR}_{\theta^\star(q)}(x, y)] - \beta \, \mathbb{E}_{x \sim \mathcal{X}} [\mathrm{KL}(q(\cdot \mid x) \,\|\, p_T(\cdot \mid x))] \tag{1}$$

subject to $\mathrm{supp}(q(\cdot \mid x)) \subseteq \{y : \mathrm{Judge}(x, y) = y^\star(x)\}$. Thus, correctness is a feasibility constraint and distillability is the optimization target; the full bilevel form is in Appendix B.1. The KL term prevents the rewriting process from inventing a new solution strategy when the teacher trace is already correct. The objective instead asks for minimally changed traces whose prefixes are easier for the learner to replay.

3.1.3 Stochastic Re-enactment

For a trajectory with step sequence $S(y) = (s_1, \ldots, s_n)$, let $z^\star(x, y) = S_{\leq k^\star(x,y)}(y)$ denote a checkpoint prefix. SRSR measures whether a learner can complete correctly from that prefix:

$$\mathrm{SRSR}_\theta(x, y) = \mathbb{E}_\omega \Big[\mathbb{I}\{\mathrm{Judge}(x, (z^\star(x, y), r_\theta(x, y; \omega))) = y^\star(x)\} \Big]. \tag{2}$$

SRSR is deliberately learner-conditioned: a prefix that is obvious to a teacher may still be weak supervision for a smaller student. Low SRSR therefore identifies checkpoints where the trace should be clarified before distillation.

3.1.4 Why Local Repair Can Matter Globally

Local repair matters because replay success compounds over future decisions. If $P(z) = \prod_{t=1}^{T} p_t$ and an edit improves a bottleneck factor p_k to p'_k, then

$$\frac{P'(z)}{P(z)} \geq \frac{p'_k}{p_k}. \tag{3}$$

The propagation extension is given in Appendix B.3. This view motivates repairing a few bottleneck prefixes rather than rewriting entire traces. It also explains why MC-RV can improve downstream behavior without replacing the teacher's successful plan: the method targets the weakest continuation points and leaves low-risk parts unchanged.

3.2 Engineering Approximation

3.2.1 From Objective to Procedure

Exact SRSR evaluation for every checkpoint is expensive, so MC-RV approximates the objective with a bottleneck-focused loop: prioritize likely fragile prefixes, rewrite locally, and validate with stochastic rollouts. This procedure is shown in Figure 2. The key engineering choice is to separate *candidate discovery* from *candidate acceptance*. Lightweight step scoring is used only to de-

cide where to spend rollout budget, while the final decision is made by re-enactment from the edited prefix. This prevents the scoring model from becoming the optimization target.

3.2.2 Step-localized Distillation Units

Given a verified-correct teacher trace y^T with prefixes $z_i^T = S_{\leq i}(y^T)$, MC-RV first converts the trace into step-localized distillation units. Each unit is represented as $(x, z_i^T, s_{i+1}^T, y_{>i+1}^T)$, where s_{i+1}^T is the transition to be inspected and $y_{>i+1}^T$ is the teacher's remaining continuation. This representation gives the rewriter enough context to preserve the teacher's plan while exposing the local state that a student must continue from. It also makes the edit auditable: after rewriting one transition, MC-RV can stitch the edited prefix back to a complete candidate trajectory and verify the final answer again.

For tool-augmented traces, the unit boundary is chosen at observable decision points such as query formulation, evidence interpretation, tool selection, and stopping decisions. These boundaries are useful because they correspond to places where a student must infer an action rather than merely copy surface text. MC-RV therefore does not treat every sentence as an equally meaningful step; it prioritizes steps that can change the future search state or the final answer. This design keeps the repair target interpretable and makes later case analysis align with the actual failure modes observed in deep-research agents.

3.2.3 Risk-guided Checkpoint Selection

We estimate each step's criticality C_i and raw expression quality Q_i using the prompts in Appendix C. Criticality measures how much later reasoning depends on the step, while raw quality measures how clearly the step exposes its premises, tool intent, parameters, and result interpretation. Since fragile checkpoints are typically both load-bearing and under-specified, we convert quality into an inverse-quality risk term and prioritize:

$$\begin{aligned} R_i &= C_i \cdot (Q_{\max} + 1 - Q_i), \\ k &\leftarrow \arg\max_{i:\, C_i \geq C_{\min}} R_i. \end{aligned} \tag{4}$$

Here $Q_{\max}$ denotes the maximum score of the raw quality rubric. This prior selects high-impact steps whose wording is likely to hide information needed for continuation. The criticality threshold filters out cosmetic or easily recoverable steps, while the inverse-quality term pushes the search toward vague queries, missing justifications, or compressed evidence transitions. Importantly, R_i only chooses the candidate bottleneck; acceptance still depends on rollout-estimated SRSR. In practice, this design avoids two failure modes: wasting validation on harmless verbose steps, and accepting fluent rewrites that do not improve continuation success.

When several checkpoints have similar risk, MC-RV evaluates them in descending R_i order until the rollout budget is exhausted or an accepted repair is found. A noisy score can change the search order, but it cannot force a rewrite into the training set without rollout evidence.

3.2.4 Rewrite and Rollout Validation

At the selected checkpoint z_k^T, MC-RV searches a local neighborhood $\mathcal{N}(x, z_k^T)$ for a rewrite that improves replay robustness while staying close to the teacher prefix:

$$z^\dagger = \arg\max_{z \in \mathcal{N}(x, z_k^T)} \widehat{\mathrm{SRSR}}_\theta(x, z) - \lambda d(z, z_k^T). \tag{5}$$

The neighborhood is intentionally narrow. Candidate rewrites may make the search intent explicit, add a missing premise, sharpen a tool query, or split an overloaded transition, but they must preserve the teacher's answer path and avoid introducing unsupported evidence. MC-RV therefore treats rewriting as local state clarification rather than free-form solution generation.

In implementation, candidate generation is conditioned on the original unit $(x, z_k^T, s_{k+1}^T, y_{>k+1}^T)$ and is instructed to edit only the selected transition. The rewriter can expose an implicit constraint, specify a more discriminative query, or add a verification action before termination, but it cannot change the target answer or rewrite unrelated earlier steps. This restriction keeps improvements attributable to local stabilization rather than a wholesale change of solution strategy.

For each candidate z, validation samples m continuations from the learner and judges whether the completed trajectory is correct:

$$\begin{gathered} \widehat{\mathrm{SRSR}}_\theta(x, z) = \frac{1}{m} \sum_{j=1}^{m} \mathbf{1}\{J_j = y^\star(x)\}, \\ J_j = \mathrm{Judge}(x, (z, u^{(j)})), \quad u^{(j)} \sim \pi_\theta(\cdot \mid x, z). \end{gathered} \tag{6}$$

We accept a rewrite only when it improves $\widehat{\mathrm{SRSR}}_\theta$ over the original checkpoint under the same roll-

out budget, preserves final-answer correctness, and remains close to the teacher prefix under the deviation penalty in Eq. (5). The original and candidate prefixes are evaluated with the same continuation model, sampling configuration, and judge, so the comparison measures the effect of the local edit rather than a change in evaluation conditions. If multiple candidates pass, MC-RV selects the candidate with the highest SRSR-adjusted score; if no candidate passes, the original trace is retained. Otherwise, the accepted prefix is stitched into a complete trajectory $y^\dagger = (z^\dagger, u^\dagger)$ and used as the SFT target. In practice, the improved training distribution is instantiated as $q(y \mid x) \approx \delta(y = y^\dagger)$: local edits approximate the distributional objective, correctness remains the constraint, and SRSR remains the selection criterion. The resulting data keep the teacher's successful reasoning structure, but expose the latent state needed for the student to continue more reliably.

4 Experiments

4.1 Experimental Setup

We use Qwen3-8B as the base model and perform supervised fine-tuning on high-quality SFT data constructed from OffSeeker's original data using MC-RV CoT-Enhancer. The teacher, the Qwen3-8B student, and the Doubao-2.0-Pro judge used for SRSR and final-answer evaluation are distinct checkpoints; the judge is not fine-tuned on MC-RV data. Detailed training procedures and hyperparameter settings can be found in Appendix A.2.

We evaluate on six deep-research benchmarks using domain-specific evaluation subsets: GAIA (Mialon et al., 2023) (Level-1, text-only, 165 questions), BrowseComp-zh/en (Zhou et al., 2025; Wei et al., 2025) (289 / 1,266 questions), HLE (Phan et al., 2025) (2,158 questions), XBench-DS (Chen et al., 2025a) (100 questions), and WebWalkerQA (Wu et al., 2025a) (680 questions). We use OffSeeker-8B as the primary reference system because it is the source of the original data and follows the same benchmark suite and evaluation protocol.

4.2 Main Results

Table 1 summarizes the results across all methods and benchmarks.

RL is Not All You Need. MC-RV-8B (SFT) achieves the top reported scores among sub-10B agents on all six benchmarks, with GAIA 49.5 (+1.9 vs. OffSeeker SFT), BC-zh 26.6 (+2.4), BC-en 11.5 (+0.9), HLE 12.5 (+0.8), XBench-DS 50.5 (+2.5), and WebWalkerQA 61.3 (+1.3). Critically, these gains are obtained without any RL interaction, showing that strong gains are possible without RL rollouts.

Competitive With Larger Models. On BrowseComp-zh (26.6), MC-RV-8B surpasses medium-scale systems such as WebSailor-32B (25.5) and approaches Claude-4-Sonnet (29.1). On WebWalkerQA (61.3), it is comparable to Claude-4-Sonnet (61.7) and slightly exceeds DeepSeek-V3.1 (61.2), despite being an 8B model trained offline.

4.3 Transfer to Math and Code Reasoning

To examine whether MC-RV transfers beyond web-agent reasoning, we further evaluate it on mathematical and code reasoning tasks using the same Qwen3-8B base model. The vanilla SFT baseline is trained on the full AM-Thinking-v1-Distilled corpus (A-M Team, 2025). For MC-RV, we keep the same full training set and training budget, but apply MC-RV enhancement only to the domain-relevant portions identified by the released category metadata: the math subset for MATH (Hendrycks et al., 2021) and the code-generation subset for LiveCodeBench v5 (Jain et al., 2024). The enhanced subset is then merged back into the full SFT corpus before training. These two settings are complementary to the deep-research benchmarks because final answers can be checked by exact matching or executable test cases rather than only by free-form LLM-as-Judge evaluation.

As shown in Table 2, MC-RV also improves SFT distillation on non-agent reasoning tasks. The gains are comparable to, and slightly larger than, the improvements observed in the deep-research setting, suggesting that the benefit of stabilizing fragile prefixes is not limited to tool-augmented web search. In math, MC-RV mainly clarifies compressed proof transitions and missing intermediate constraints; in code, it tends to expose edge cases, input-output conditions, and verification steps that are implicit in the original solution trace.

4.4 Analysis

4.4.1 SRSR before/after rewriting.

We further report direct before/after measurements of Step-wise Replay Success Rate (SRSR) on the accepted rewritten trajectories. Under the same

Method	GAIA	BC-zh	BC-en	HLE	XBench-DS	WebWalkerQA
Large Size Foundation Models (>100B)						
OpenAI-o3	70.5	58.1	49.7	20.2	66.7	71.7
Claude-4-Sonnet	68.3	29.1	12.2	20.3	64.6	61.7
DeepSeek-V3.1	63.1	49.2	30.0	29.8	71.0	61.2
DeepSeek-V3.2	63.5	47.9	40.1	27.2	71.0	–
Kimi-K2	57.7	28.8	14.1	18.1	50.0	63.0
Medium Size Deep Research Agents (30B–72B)						
MiroThinker-32B-DPO-v0.1	57.3	17.0	13.0	11.8	–	49.3
ASearcher-Web-QwQ	52.8	15.6	5.2	12.5	42.1	34.3
WebDancer-QwQ	51.5	18.0	3.8	–	39.0	47.9
WebSailor-32B	53.2	25.5	10.5	–	53.3	–
WebSailor-72B	55.4	30.1	12.0	–	55.0	–
WebSailor-v2-30B (SFT)	66.0	28.3	24.4	23.9	–	–
WebSailor-v2-30B (RL)	74.1	44.1	35.3	30.6	–	–
WebShaper-32B	52.4	–	–	–	–	51.4
WebShaper-72B	60.0	–	–	–	–	52.2
Tongyi-DeepResearch	70.9	46.7	43.4	32.9	–	72.2
Small Size Deep Research Agents (<10B)						
MiroThinker-8B-DPO-v0.1	46.6	13.6	8.7	–	–	45.7
WebSailor-7B	33.0	14.2	6.7	–	34.3	–
DeepDive-9B (SFT)	–	15.7	5.6	–	35.0	–
DeepDive-9B (RL)	–	15.1	6.3	–	38.0	–
OffSeeker-8B (SFT)	47.6	24.2	10.6	11.7	48.0	60.0
Random-Rewrite-SFT†	48.5	25.3	10.9	12.0	48.7	60.4
Quality-Only-SFT†	48.5	26.0	11.2	12.2	49.7	60.9
MC-RV-8B (SFT)†	**49.5**	**26.6**	**11.5**	**12.5**	**50.5**	**61.3**
vs. OffSeeker (SFT)	+1.9	+2.4	+0.9	+0.8	+2.5	+1.3

Table 1: Main results on the evaluation subsets defined in Section 4.1. All scores of MC-RV are computed as Pass@1 using the Doubao-2.0-Pro LLM-as-Judge protocol. GAIA denotes the Level-1 text-only GAIA subset; BC-zh and BC-en denote BrowseComp-ZH and BrowseComp, respectively; XBench-DS refers to XBench-DeepSearch. Scores marked with † are from our own tests; others are from official reports or published deep-research works. **Bold** = best among Small Size Deep Research Agents (<10B).

Domain	Optimized subset	Benchmark	SFT	MC-RV
Math	AM-Thinking-Math	MATH (5,000)	82.6	**84.9** (+2.3)
Code	AM-Thinking-Code	LCB-v5 (880)	65.9	**68.5** (+2.6)

Table 2: Transfer results on math and code reasoning with the same Qwen3-8B base model. Both SFT and MC-RV are trained on the full AM-Thinking-v1-Distilled corpus; MC-RV differs only by enhancing the indicated domain subset before merging it back into the full SFT set.

Subset	Original SRSR	Rewritten SRSR
All trajectories	55.7%	64.9%
Trajectories with fragile steps	43.6%	55.1%

Table 3: Before/after SRSR comparison. MC-RV yields larger gains on trajectories with fragile steps.

experiment settings, the average SRSR increases from **55.7%** for the original traces to **64.9%** after MC-RV rewriting, corresponding to an absolute gain of **9.2** points and a relative improvement of **16.5%** (Table 3). These results directly support the mechanism proposed in Section 3: MC-RV improves the replay robustness of verified-correct trajectories, rather than merely rewriting them into superficially cleaner forms.

Notably, the improvement is especially pronounced on fragile steps. In our analysis, **21.7%** of turns are identified as fragile under the SRSR criterion (SRSR < 0.5; see Appendix A.3), suggesting that local instability is fairly common even among verified-correct trajectories. On this subset, the average SRSR increases from **43.6%** to **55.1%**, yielding a larger absolute gain of **11.5** points. This pattern is consistent with the design goal of MC-RV: rather than uniformly rewriting all steps, it focuses on bottleneck prefixes whose continuation instability most strongly limits replay robustness.

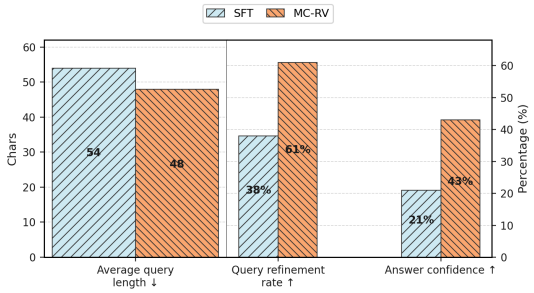

Figure 3: Search Behavior on GAIA and BrowseComp-zh across incremental components.

4.4.2 Search Behavior Analysis

To understand whether replay-robust supervision changes the agent's process, not only its final answer, we compare search behavior statistics between the SFT baseline and MC-RV at inference time:

- **Average query length:** SFT baseline: 54 chars; MC-RV: 48 chars, indicating shorter and more targeted queries.
- **Query refinement rate:** SFT: 38%; MC-RV: 61%, showing more follow-up search after partial evidence.
- **Answer confidence:** MC-RV uses explicit self-verification in 43% of correct trajectories vs. 21% for SFT.

These differences align with the stabilization component: MC-RV teaches more targeted search, more refinement, and more explicit verification, suggesting that gains are not merely outcome-level improvements.

4.4.3 Case Study

We inspect representative rewritten trajectories to understand what MC-RV changes at the step level. Figure 4 summarizes two cases, with fuller descriptions in Appendix A.2.

Both cases show that endpoint-correct traces can still expose unstable continuation states. MC-RV repairs the bottleneck prefix rather than globally rewriting the trajectory, supporting the same mechanism as the SRSR results.

4.4.4 Ablation Study

Table 4 reports a component-level ablation averaged over the six deep-research benchmarks.

Case A: Search bottleneck in a verified-correct BrowseComp trace
Correct answer, unstable search state

Question type. Find a specific clinical-trial result hidden behind many similarly named medical pages.

Fragile prefix. The trace starts from the broad query *clinical trial cancer treatment 2020*.

Rollout symptom. Continuations often drift to unrelated trials or collect generic background evidence before reaching the target paper.

⇓ **replay-guided local repair** ⇓

MC-RV repair. Rewrite the first action to the entity-grounded query *pembrolizumab KEYNOTE-042 OS benefit 2020*.

Supported contribution. MC-RV localizes and repairs the hidden search bottleneck controlling replay robustness.

Case B: Stopping bottleneck in a verified-correct GAIA trace
Correct answer, unstable stopping state

Question type. Answer an institution-history question where secondary and primary sources disagree.

Fragile prefix. The trace has a plausible secondary-source date and is one action away from finishing.

Rollout symptom. Continuations often stop with the approximate date instead of checking the official history page.

⇓ **replay-guided local repair** ⇓

MC-RV repair. Insert an official-history verification step before producing the final answer.

Supported contribution. MC-RV distinguishes answer correctness from process stability and repairs the brittle stopping prefix.

Figure 4: Two single-column case studies showing the mechanism of MC-RV. Each original trace is endpoint-correct but has a fragile prefix; MC-RV repairs that prefix while preserving the answer target.

Variant	Avg Acc.	Correct / 4,658
SFT only	33.7	992
Rewrite-only (random step)	34.3 (+0.6)	1,010
Quality-only selection	34.8 (+1.1)	1,024
Criticality-only selection	34.7 (+1.0)	1,021
Risk prior, no MC validation	35.0 (+1.3)	1,034
Full MC-RV	**35.3 (+1.6)**	**1,043**

Table 4: Fine-grained ablation over six benchmarks. The last column pools correct answers over the 4,658 questions listed in Section 4.1.

The component ablations disentangle the gains: quality-only and criticality-only selection reach **34.8** and **34.7**, their risk-prior combination reaches **35.0**, and adding MC validation gives the full **35.3**, showing that both bottleneck selection and replay-based acceptance matter.

5 Conclusion

This paper shows that endpoint-correct CoT traces are not always good SFT targets. *MC-RV CoT-Enhancer* identifies fragile verified-correct prefixes, repairs them only when replay validation improves SRSR while preserving correctness, and improves strong 8B baselines across deep-research, math, and code benchmarks. The results suggest reasoning supervision should be judged by resumability, not only final correctness.

6 Limitations

This work has several limitations. First, because the framework relies on judge-based evaluation and rewrite acceptance, it may inherit evaluator biases and preferentially reinforce reasoning patterns that appear more stable under the chosen judge without necessarily being more reliable in all settings.

Second, our framework is intentionally restricted to the verified-correct solution manifold. While this keeps optimization focused on stabilizing correct-but-brittle trajectories, it does not address how to repair incorrect teacher traces.

Third, although we add transfer experiments on mathematical and code reasoning, our experiments still use a single primary student scale and model family. Broader validation across additional model families, scale regimes, and reasoning domains remains important future work.

Finally, our current implementation uses judge-based success estimation and local rewrite acceptance rules that only approximate the underlying objective of distillability. Developing cheaper and more principled estimators of prefix-conditioned learnability remains an important direction for future work.

References

A-M Team. 2025. AM-Thinking-v1-Distilled. https://huggingface.co/datasets/a-m-team/AM-Thinking-v1-Distilled.

Maria-Florina Balcan, Avrim Blum, Zhiyuan Li, and Dravyansh Sharma. 2025. On learning verifiers and implications to chain-of-thought reasoning. *arXiv preprint arXiv:2505.22650*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, and 1 others. 2025a. xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations. *arXiv preprint arXiv:2506.13651*.

Wenqing Chen, Weicheng Wang, Zhixuan Chu, Kui Ren, Zibin Zheng, and Zhichao Lu. 2024. Self-para-consistency: Improving reasoning tasks at low cost for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14162–14167, Bangkok, Thailand. Association for Computational Linguistics.

Xinghao Chen, Zhijing Sun, Guo Wenjin, Miaoran Zhang, Yanjun Chen, Yirong Sun, Hui Su, Yijie Pan, Dietrich Klakow, Wenjie Li, and Xiaoyu Shen. 2025b. Unveiling the key factors for distilling chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15094–15119, Vienna, Austria. Association for Computational Linguistics.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2023. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*.

Jishnu Ray Chowdhury and Cornelia Caragea. 2025. Zero-shot verification-guided chain of thoughts. *arXiv preprint arXiv:2501.13122*.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578, Bangkok, Thailand. Association for Computational Linguistics.

Cong Thanh Do, Rama Sanand Doddipatla, and Kate Knill. 2025. Effectiveness of chain-of-thought in distilling reasoning capability from large language models. In *Proceedings of the 18th International Natural Language Generation Conference*, pages 833–845, Hanoi, Vietnam. Association for Computational Linguistics.

Austin Feng, Marius Alonso, and Ambroise Odonnat. 2025. Optimal self-consistency for efficient reasoning with large language models. *arXiv preprint arXiv:2511.12309*.

Vatsal Gupta, Pranshu Pandya, Tushar Kataria, Vivek Gupta, and Dan Roth. 2024. Evaluating concurrent robustness of language models across diverse challenge sets. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22162–22184, Miami, Florida, USA. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arber, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for*

Computational Linguistics: ACL 2023, pages 8003–8017, Toronto, Canada. Association for Computational Linguistics.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Zeyu Jia, Alexander Rakhlin, and Tengyang Xie. 2025. Do we need to verify step by step? rethinking process supervision from a theoretical perspective. *arXiv preprint arXiv:2502.10581*.

Hyunseok Lee, Seunghyuk Oh, Jaehyung Kim, Jinwoo Shin, and Jihoon Tack. 2025. ReVISE: Learning to refine at test-time via intrinsic self-verification. *arXiv preprint arXiv:2502.14565*.

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

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. 2025. S^2R: Teaching LLMs to self-verify and self-correct via reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22632–22654, Vienna, Austria. Association for Computational Linguistics.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.

Lujie Niu, Haochen Sun, Fangkun Zhao, Sheng Chen, Zimeng Bai, Jiawei Zhang, Caixia Yuan, and Xiaojie Wang. 2025. CoTD-PO: Chain-of-thought distillation with preference optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 19975–19986, Suzhou, China. Association for Computational Linguistics.

Debjit Paul, Robert West, Antoine Bosselut, and Boi Faltings. 2024. Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15012–15032, Miami, Florida, USA. Association for Computational Linguistics.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, and 1 others. 2025. Humanity's last exam. *arXiv preprint arXiv:2501.14249*.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, Pranav Sandeep Dulepet, Saurav Vidyadhara, Dayeon Ki, Sweta Agrawal, Chau Minh Pham, Gerson C. Kroiz, Feileen Li, Hudson Tao, Ashay Srivastava, and 12 others. 2024. The prompt report: A systematic survey of prompting techniques. *arXiv preprint arXiv:2406.06608*.

Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. 2023. Algorithm of thoughts: Enhancing exploration of ideas in large language models. *arXiv preprint arXiv:2308.10379*.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2025. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *The Thirteenth International Conference on Learning Representations*.

Akshit Sinha, Arvindh Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. 2025. The illusion of diminishing returns: Measuring long horizon execution in llms. *arXiv preprint arXiv:2509.09677*.

Xiaoyu Tian, Sitong Zhao, Haotian Wang, Shuaiting Chen, Yiping Peng, Yunjie Ji, Han Zhao, and Xiangang Li. 2025. DeepDistill: Enhancing LLM reasoning capabilities via large-scale difficulty-graded data training. *arXiv preprint arXiv:2504.17565*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. 2025. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He,

Deyu Zhou, Pengjun Xie, and 1 others. 2025a. Webwalker: Benchmarking llms in web traversal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10290–10305.

Tong Wu, Chong Xiang, Jiachen T Wang, G Edward Suh, and Prateek Mittal. 2025b. Effectively controlling reasoning models through thinking intervention. *arXiv preprint arXiv:2503.24370*.

Xin Xu, Shizhe Diao, Can Yang, and Yang Wang. 2025. Can we verify step by step for incorrect answer detection? In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pages 8322–8330, Montreal, Canada. International Joint Conferences on Artificial Intelligence Organization.

Zhaohui Yang, Chenghua He, Xiaowen Shi, Shihong Deng, Linjing Li, Qiyue Yin, and Daxin Jiang. 2025. Beyond the first error: Process reward models for reflective mathematical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 4711–4728, Suzhou, China. Association for Computational Linguistics.

Fei Yu, Yingru Li, and Benyou Wang. 2025. Scaling flaws of verifier-guided search in mathematical reasoning. *arXiv preprint arXiv:2502.00271*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. STaR: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. 2024. Chain of preference optimization: Improving chain-of-thought reasoning in llms. *Advances in Neural Information Processing Systems*, 37:333–356.

Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, and 1 others. 2025. Browsecomp-zh: Benchmarking web browsing ability of large language models in chinese. *arXiv preprint arXiv:2504.19314*.

Zhanke Zhou, Rong Tao, Jianing Zhu, Yiwen Luo, Zengmao Wang, and Bo Han. 2024. Can language models perform robust reasoning in chain-of-thought prompting with noisy rationales? *Advances in Neural Information Processing Systems*, 37:123846–123910.

A Experiment Details

A.1 Artifact Licenses and Intended Use.

We use publicly available research artifacts, including Qwen3-8B, OffSeeker data, GAIA, BrowseComp/BrowseComp-zh, HLE, XBench-DS, WebWalkerQA, and LLaMA-Factory, following their respective licenses, terms of use, and citation requirements. These artifacts are used only for research purposes, including model initialization, supervised fine-tuning data construction, and benchmark evaluation. We do not redistribute third-party benchmark questions, hidden labels, official test sets, or raw data from existing artifacts. Any artifacts created by this work, including rewritten reasoning traces, code, or trained checkpoints, are intended for research use and will be released only in forms compatible with the corresponding upstream licenses and access conditions.

A.2 Case Study

Case A: Search bottleneck in a verified-correct BrowseComp trace. The question asks for a specific clinical-trial result, but many superficially similar pages mention cancer treatment, trial years, and outcome terminology. The original teacher trace eventually reaches the correct answer, yet its first search action is a weak supervision prefix: `clinical trial cancer treatment 2020`. This query is locally plausible, so a correctness filter would not reject the trajectory. Under stochastic re-enactment, however, continuations from this prefix often branch into unrelated trials or collect generic background evidence before finding the target paper. MC-RV therefore treats the first search step as a high-risk checkpoint and rewrites it into an entity-grounded query: `pembrolizumab KEYNOTE-042 OS benefit 2020`. The edit preserves the same answer target, but makes the next evidence state much more constrained. This case supports the core claim that endpoint correctness can hide weak search prefixes: the trace is correct, but the original prefix is not reliably distillable.

Case B: Stopping bottleneck in a verified-correct GAIA trace. The question asks for an institution-history fact whose value appears in both secondary summaries and the institution's official history page. The original trace reaches a plausible secondary-source date and approaches the final `finish` action. As a complete trajectory, it can still end with the correct answer; as process supervision, the prefix is fragile because it does not make the need for primary-source verification explicit. Rollouts from this state often terminate early with the secondary-source date, especially when stochastic decoding favors shorter completion. MC-RV localizes the checkpoint around the premature stopping decision and rewrites it

by adding a verification query to the official history page before the final answer. The accepted rewrite keeps the answer target unchanged while changing the process state: the student now learns that a plausible secondary-source answer must be checked before termination. This case supports the step-localized design of MC-RV: the useful edit is a targeted repair of the bottleneck prefix, not a global restatement of the whole trajectory.

A.3 Fragility Distribution

Figure 5 reports the turn-level distribution used to contextualize the SRSR analysis in Section 4. The figure supports the same mechanism-level claim as Table 3: a nontrivial fraction of steps in verified-correct traces are locally fragile, so endpoint filtering alone does not guarantee that the trace is stable supervision for distillation.

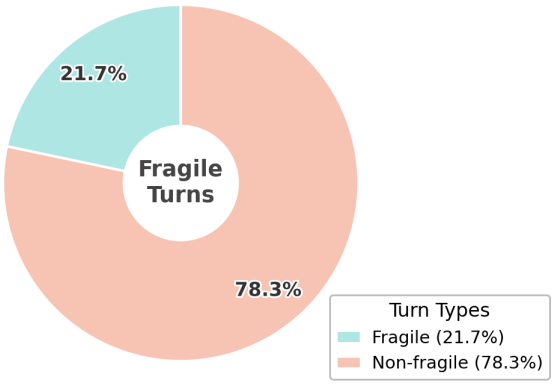

Figure 5: Turn-level fragility distribution under the SRSR criterion. We define fragile turns as steps with SRSR < 0.5; 21.7% of turns fall into this category.

A.4 Model Training Details

For supervised fine-tuning, we set the maximum sequence length to 128k tokens to accommodate long-horizon reasoning and multi-step tool interactions. To enable efficient long-context training, we adopt 360-LLaMA-Factory with sequence parallelism. We use the `cosine_with_min_lr` learning rate scheduler, with an initial learning rate of 3e−5, a minimum learning rate of 1e−7, and a warmup ratio of 0.1. The model is trained for 3 epochs, with global training batch size of 512.

We determine all the hyperparameters by grid search. All training experiments are conducted on 64 NVIDIA H20 GPUs, each with 80 GB of memory.

A.5 Preprocessing Cost

MC-RV is an offline data-construction procedure and adds no inference-time cost to the trained student. For SRSR estimation, we use $M = 8$ stochastic continuations per evaluated prefix. After lightweight criticality and quality scoring, we validate only the top $K = 3$ candidate checkpoints in each trajectory. For each checkpoint, at most two local rewrite candidates are evaluated together with the original prefix, so the worst-case validation budget is $3 \times (2+1) \times 8 = 72$ continuation rollouts per trajectory. In practice, early stopping after an accepted rewrite reduces the average validation cost to 27.6 continuations per processed trajectory. This cost is incurred once when constructing the SFT data and is independent of the downstream evaluation benchmarks.

B Additional Theoretical Details

B.1 Full Bilevel Formulation

In the main text, we write $\theta^\star(q)$ as the student obtained by fitting π_θ to the rewritten trajectory distribution q. The full formulation is a constrained bilevel problem:

$$\begin{aligned} \max_{q \in \mathcal{Q}} \quad & \mathbb{E}_{x \sim \mathcal{X}} \mathbb{E}_{y \sim q(\cdot \mid x)} \big[\mathrm{SRSR}_{\theta^\star(q)}(x, y) \big] \\ & - \beta \, \mathbb{E}_{x \sim \mathcal{X}} \big[\mathrm{KL}\big(q(\cdot \mid x) \,\|\, p_T(\cdot \mid x) \big) \big] \\ \text{s.t.} \quad & \theta^\star(q) = \arg\min_\theta \\ & \qquad \mathbb{E}_{x \sim \mathcal{X}} \big[\mathrm{KL}\big(q(\cdot \mid x) \,\|\, \pi_\theta(\cdot \mid x) \big) \big], \\ & \mathrm{supp}\big(q(\cdot \mid x) \big) \subseteq \\ & \qquad \big\{ y : \mathrm{Judge}(x, y) = y^\star(x) \big\}. \end{aligned} \tag{7}$$

The upper-level objective seeks a rewritten distribution that improves prefix-conditioned replay robustness while remaining close to the teacher-induced distribution p_T. The support constraint enforces correctness: only verified-correct trajectories are admissible.

The lower-level problem defines the student induced by q. Since

$$\begin{aligned} \mathrm{KL}\big(q(\cdot \mid x) \,\|\, \pi_\theta(\cdot \mid x) \big) = H\big(q(\cdot \mid x) \big) \\ + \mathbb{E}_{y \sim q(\cdot \mid x)} [- \log \pi_\theta(y \mid x)]. \end{aligned} \tag{8}$$

and the entropy term does not depend on θ, the inner problem is equivalent to standard likelihood fitting:

$$\theta^\star(q) = \arg\min_\theta \mathbb{E}_{x \sim \mathcal{X}} \mathbb{E}_{y \sim q(\cdot \mid x)} [- \log \pi_\theta(y \mid x)]. \tag{9}$$

Thus, the formulation does not assume a special student-training rule; it simply makes explicit that changing the training-trace distribution q changes the student that distills from it.

B.2 Checkpoint Prefixes, Rollouts, and Monte Carlo Estimation

Let y be a verified-correct trajectory for input x, and let

$$S(y) = (s_1, \ldots, s_n)$$

denote its step sequence. A checkpoint prefix is a partial reasoning state

$$z^\star(x, y) \triangleq S_{\leq k^\star(x,y)}(y),$$

where $k^\star(x, y) \in \{1, \ldots, n\}$ is a selected checkpoint index. In the theory, $k^\star(x, y)$ is left abstract; in the implementation, it is instantiated by the step-selection mechanism described in Section 3.2.

Given a student policy π_θ, we define a stochastic continuation from the checkpoint as

$$r_\theta(x, y; \omega) := \mathrm{Rollout}(\pi_\theta \mid x, z^\star(x, y); \omega), \quad (10)$$

where ω denotes the randomness introduced by stochastic decoding (e.g., sampling noise, temperature-based branching, or other controlled perturbations).

The **stochastic re-enactment success rate** (SRSR) is then

$$\mathrm{SRSR}_\theta(x, y) = \mathbb{E}_\omega\Big[\mathbb{I}\Big\{\mathrm{Judge}\Big(x,\ \big(z^\star(x, y),\ r_\theta(x, y; \omega)\big)\Big) = y^\star(x)\Big\}\Big]. \quad (11)$$

This quantity is prefix-conditioned and learner-dependent. It asks whether a learner, resumed from an intermediate reasoning state, can still complete to the verified-correct solution under stochastic continuation.

In practice, SRSR is estimated by Monte Carlo rollouts. Given M i.i.d. samples $\omega_1, \ldots, \omega_M$, we use

$$\widehat{\mathrm{SRSR}}_\theta^{(M)}(x, y) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{I}\Big\{\mathrm{Judge}\Big(x,\ \big(z^\star(x, y),\ r_\theta(x, y; \omega_m)\big)\Big) = y^\star(x)\Big\}. \quad (12)$$

When needed, a checkpoint can be classified as fragile by thresholding the estimated replay success rate:

$$\widehat{\mathrm{SRSR}}_\theta^{(M)}(x, y) < \tau, \quad (13)$$

where τ is an analysis-time threshold. In our empirical analysis, we use $\tau = 0.5$.

B.3 Log-risk Interpretation and Propagation Bound

This subsection expands the bottleneck interpretation in the main text and shows how a local replay improvement can induce multiplicative gains in end-to-end continuation reliability.

Let z be a checkpoint prefix, and let completion from z require T future decisions. Write

$$P(z) = \prod_{t=1}^{T} p_t, \qquad p_t \in (0, 1),$$

where p_t is the conditional success probability at future step t under the relevant rollout dynamics. Define the step-wise log-risk

$$r_t \triangleq -\log p_t, \quad \text{so that} \quad P(z) = \exp\Big(-\sum_{t=1}^{T} r_t\Big). \quad (14)$$

Now suppose a local edit is made at bottleneck step k, changing the local success probability from p_k to $p'_k > p_k$. Define

$$\delta \triangleq \log\Big(\frac{p'_k}{p_k}\Big) > 0. \quad (15)$$

No propagation. Assume first that only the local factor changes and all later factors remain unchanged:

$$p'_t = p_t, \qquad t \neq k.$$

Then

$$\frac{P'(z)}{P(z)} = \frac{p'_k}{p_k} = \exp(\delta). \quad (16)$$

If, in addition, the edit improves the local factor by at least an additive margin $\Delta > 0$, i.e.,

$$p'_k \geq p_k + \Delta,$$

then

$$\frac{P'(z)}{P(z)} = \frac{p'_k}{p_k} \geq 1 + \frac{\Delta}{p_k}. \quad (17)$$

This already shows a key asymmetry: improving a fragile factor with small p_k can produce a relatively large multiplicative change in end-to-end replay success.

With downstream propagation. The more interesting case is when the repaired step also clarifies downstream reasoning. To model this, assume the edit affects an effective dependency window of length L_k, where $L_k \in \{0, 1, \ldots, T - k\}$, and let the downstream transmission strength be

$\eta \in [0, 1]$. We impose the following risk-reduction condition:

$$r'_t \le r_t - \eta\,\delta, \qquad t = k+1, \ldots, k+L_k. \quad (18)$$

At the edited step itself,

$$r'_k = r_k - \delta. \quad (19)$$

For all other steps outside the affected region, we assume $r'_t = r_t$.

Then

$$\log\Big(\frac{P'(z)}{P(z)}\Big) = \sum_{t=1}^{T}(r_t - r'_t) \quad (20)$$

$$\ge (r_k - r'_k) + \sum_{t=k+1}^{k+L_k}(r_t - r'_t) \quad (21)$$

$$\ge \delta + L_k\,\eta\,\delta \quad (22)$$

$$= (1 + \eta L_k)\delta. \quad (23)$$

Exponentiating both sides yields

$$\frac{P'(z)}{P(z)} \ge \exp\big((1+\eta L_k)\delta\big) = \Big(\frac{p'_k}{p_k}\Big)^{1+\eta L_k}. \quad (24)$$

If $p'_k \ge p_k + \Delta$, then

$$\frac{P'(z)}{P(z)} \ge \Big(1 + \frac{\Delta}{p_k}\Big)^{1+\eta L_k}. \quad (25)$$

Equation (24) recovers the local-only case when $\eta = 0$ or $L_k = 0$, and becomes strictly stronger when downstream clarification reduces future replay risk as well. The role of this derivation is interpretive rather than axiomatic: it formalizes the intuition that replay robustness may be dominated by a few bottleneck checkpoints, and that repairing them can yield amplified end-to-end gains.

B.4 Scope and Relation to Other Notions

Our formulation is intentionally narrower than general reasoning correction. It operates inside the space of *verified-correct* trajectories and asks a different question: among trajectories that already end correctly, which ones provide better supervision for a student that must resume from intermediate prefixes?

This differs from several nearby notions.

Endpoint correctness. Endpoint filtering only asks whether a completed trajectory reaches the correct final answer. It does not distinguish a stable trajectory from a brittle one if both end correctly.

Step-wise correctness scoring. A step can be locally correct while still leaving the learner in a poorly specified state for continuation. SRSR is therefore not a substitute for step verification; it measures a different object, namely, *prefix-conditioned replay usability*.

Whole-trajectory pass rate. A model may achieve a high pass rate when decoding from scratch, yet still fail to continue reliably from an intermediate prefix. SRSR isolates this continuation aspect by conditioning explicitly on a checkpoint.

Incorrect-trajectory repair. Our theory does not model arbitrary correction of wrong trajectories. The support constraint in Eq. (7) keeps optimization inside the verified-correct solution manifold. The goal is not to fix wrong answers, but to stabilize correct-but-brittle supervision signals.

In this sense, Step-wise Fragility captures the gap between *being correct in hindsight* and *being reliably teachable through partial-prefix continuation*, which is the property relevant to distillation.

C Scoring Prompts

C.1 Criticality Assessment

Identifier: `STEP_CRITICALITY_ASSESSMENT_PROMPT`

```
STEP_CRITICALITY_ASSESSMENT_PROMPT = """
**Task: Assess how critical a reasoning step is to
     the overall solution.**

**Context:**
You are identifying "load-bearing" steps in a tool-
     augmented reasoning process.
These are steps that, if written vaguely or
     incorrectly, would make the entire solution
     fail.

**Special Considerations for Tool Calls:**
- Tool calls that gather essential information are
     highly critical
- The reasoning that leads to a tool call is often
     critical
- Analysis of tool results that directly affects the
      conclusion is critical

**Evaluation Context:**
1. **Problem Description:**
   {problem_text}

2. **All Steps:**
   {all_steps}

3. **Current Step to Evaluate (Step {step_id}):**
   {current_step_text}

**Criticality Dimensions:**
1. **Information Gathering**: Does this step gather
     essential information for solving the problem?
2. **Decision Point**: Does this step make a key
     decision that affects subsequent steps?
3. **Synthesis**: Does this step synthesize
     information to form the answer?

**Scoring:**
- 10 (Critical): This step is essential - removing
     or changing it would break the solution
- 8-9 (Very Important): Major milestone in the
     reasoning process
```

```
- 6-7 (Important): Necessary but more mechanical
- 4-5 (Moderate): Helpful but not central
- 1-3 (Low): Cosmetic or easily recoverable

**Output Requirement:**
Output ONLY a valid JSON object:

{{
  "criticality_score": 8,
  "comment": "This step performs the key search that
      retrieves essential information for answering
     the question."
}}

Your response:
"""
```

C.2 Quality Assessment

Identifier: `STEP_QUALITY_ASSESSMENT_PROMPT`

```
STEP_QUALITY_ASSESSMENT_PROMPT = """
**Task: Evaluate the EXPRESSION QUALITY of a tool-
     augmented reasoning step.**

**Context:**
You are judging how CLEAR, EXPLICIT, and WELL-
     REASONED this step is.
For tool calls, evaluate whether the reasoning for
     making the call is clear and the parameters are
      appropriate.

**Evaluation Context:**
1. **Problem Description:**
   {problem_text}

2. **Previous Steps:**
   {previous_steps}

3. **Current Step to Evaluate (Step {step_id}):**
   {current_step_text}

**Quality Dimensions:**
1. **Clarity**: Is the reasoning clear and easy to
     follow?
2. **Justification**: For tool calls, is there clear
      reasoning for why this tool is being used?
3. **Parameter Quality**: For tool calls, are the
     parameters well-chosen and appropriate?
4. **Result Interpretation**: Is the interpretation
     of results accurate and complete?

**Scoring Standards:**
- **10 (Perfect)**: Exceptionally clear, well-
     justified, optimal parameters
- **8-9 (Excellent)**: Very clear with minor room
     for improvement
- **6-7 (Good)**: Basically clear but some gaps in
     reasoning
- **4-5 (Average)**: Understandable but vague or
     missing justification
- **2-3 (Poor)**: Confusing or poorly justified
- **1 (Bad)**: Very unclear or inappropriate

**Output Requirement:**
Output ONLY a valid JSON object:

{{
  "score": 6,
  "comment": "The tool call is appropriate but the
     reasoning for choosing this specific search
     query is not explained..."
}}

Your response:
"""
```

D AI Assistance Statement

We used generative AI tools, including ChatGPT, as auxiliary support during manuscript preparation and experimental organization. Their use was limited to language polishing, grammar and clarity editing, formatting suggestions, code-command drafting, log summarization, and consistency checks for reported tables and figures. These tools did not independently generate research ideas, select data, run experiments, produce reported results, or determine the paper's scientific claims. All experimental outputs were obtained from author-executed code. All analyses, numerical results, and final text were checked and approved by the authors, who take full responsibility for the content of this paper.