Mini Artichokes: Reliable Self-Correction from Validated Overlap

Hyeongbin Park*

February 14, 2026

Abstract

We study a practical self-correction setting in which an LLM is allowed to call an API repeatedly but receives *no* external feedback beyond its own verification and critique prompts (no tools, no internet, no human-in-the-loop). While "garbage in, garbage out" remains a strong prior for iterative prompting, we find that carefully *filtering* what the model learns from its own critiques can materially improve reliability on some hard reasoning items.

We introduce **Mini Artichokes**, an API-only verification-and-refinement loop that extracts the *overlap* between two independent diagnoses of a failure and then validates that overlap before applying a correction. The key idea is to reduce self-correction drift by only feeding back a small, high-agreement error signature. We evaluate on a heterogeneous set of Korean and mathematics benchmarks (PSAT Linguistic Logic, LEET Analytical Reasoning subset, Putnam 2024, AIME 2026). Overall accuracy gains are modest when averaged over entire suites, but on PSAT the gains are most visible in *coverage*: under a best-of-20 view (count an item if it is solved at least once over 20 trials; 2.5 points per item), pass@1 reaches 62.5/100 (25/40 items) while Mini Artichokes reaches 92.5/100 (37/40 items). Crucially, Mini Artichokes solves 12 of the 15 PSAT items that pass@1 never solves in 20 trials, and we do not observe the reverse (a PSAT item solved at least once by pass@1 but never by Mini Artichokes), suggesting it is best used as a fallback module for hard, low-confidence failures rather than a wholesale replacement for pass@1.

1 Introduction

Iterative "self-correction" is often described as a free lunch: ask the model to critique itself, apply fixes, and repeat until the answer stabilizes. In practice, the critique channel is noisy and the loop can amplify spurious issues, especially without external signals (tests, tools, humans). This tension is an instance of the community's long-standing "garbage in, garbage out" heuristic: if the loop's inputs are low-quality, more iterations can simply produce more confident noise.

This paper focuses on a constrained but realistic regime: *pure API repetition*. We intentionally exclude tools and external feedback to (i) keep the architecture model-agnostic—as API intelligence improves, the same loop can get stronger without redesign—and (ii) preserve composability, i.e., Mini Artichokes can be embedded as a subroutine inside larger agentic systems.

Name. The name *Mini Artichokes* is deliberate. An artichoke has many layers; likewise, our loop peels through multiple refinement layers. "Mini" emphasizes that the method is meant to be a small component that can plug into a larger system (where external tools or validators may exist), rather than a monolithic agent.

*Department of Biomedical Engineering, Yonsei University, Mirae Campus, Wonju-si, Gangwon-do, Republic of Korea. hbin@yonsei.ac.kr

Contributions.

- **Validated overlap correction.** We propose an overlap-and-validate mechanism that filters critiques to the intersection of two diagnoses before updating the solution.
- **Budgeted, API-only loop.** We show how to run the architecture under explicit call budgets (e.g., `rcap35` for cost control) while remaining tool-free and modular.
- **Hard-item reporting.** We report results in a way that exposes where the architecture matters most: items that are rarely solved by pass@1 but become frequently solvable under Mini Artichokes, while including full per-item tables in the appendix for transparency.

2 Related Work

Mini Artichokes is most directly inspired by model-agnostic verification-and-refinement pipelines for hard reasoning [3], and by open-source implementations of iterative refinement patterns [6]. We also adopt the pass-streak acceptance rule and fail-streak early stopping heuristic from the IMO25 pipeline [3, 11]. We adapt these ideas to a setting that emphasizes (a) conservative feedback filtering and (b) API-only deployability for integration into larger systems.

3 Mini Artichokes

3.1 Overview

Mini Artichokes decomposes the loop into roles that can be implemented via prompting:

1. **Solver** proposes an answer (and optionally a solution).
2. **Verifier** checks the proposal. If it passes, we accept after a short pass-streak.
3. If verification fails: two independent **Diagnosers** describe why the proposal is wrong.
4. **Overlap extractor** computes the shared core between the two diagnoses (the candidate "error signature").
5. **Overlap validator** checks that the signature actually explains the failure (and is not hallucinated).
6. **Corrector** applies a targeted fix conditioned on the validated signature.

The *validated overlap* step is the main novelty: instead of feeding an unconstrained critique back into the solver, we filter to what two critiques agree on *and* require the model to justify that the overlap is relevant. This reduces prompt drift and helps avoid "fixing" imaginary issues.

Solver quality floor (ICR inner loop). In our implementation, the **Solver** is not a single-shot API call. Instead, we run a small ICR-style inner loop (critique → revise) inside SOLVE to secure an *intelligence floor*: if the initial draft is too low-quality, downstream verification and overlap filtering have little signal to work with, and the probability of ever reaching a correct answer collapses. We tested multiple alternatives for this "floor" (single-shot solves, longer prompts, different role prompts), and the ICR inner loop was the most consistently reliable under similar cost budgets [6].

Algorithm 1 Mini Artichokes (validated overlap verification-and-refinement)

Require: problem x, max rounds M, pass streak p, fail streak f, per-round cap `rcap`

$y \leftarrow \textsc{Solve}(x)$
$passes \leftarrow 0$; $fails \leftarrow 0$
for $r = 1$ to M **do**
  $(ok, report) \leftarrow \textsc{Verify}(x, y)$
  if ok **then**
    $passes \leftarrow passes + 1$; $fails \leftarrow 0$
    if $passes \geq p$ **then**
      return y
    end if
  else
    $fails \leftarrow fails + 1$; $passes \leftarrow 0$
    $d_1 \leftarrow \textsc{Diagnose}(x, y, report)$
    $d_2 \leftarrow \textsc{Diagnose}(x, y, report)$ ▷ independent
    $o \leftarrow \textsc{ExtractOverlap}(d_1, d_2)$
    $o \leftarrow \textsc{ValidateOverlap}(x, y, report, o)$
    $y \leftarrow \textsc{Correct}(x, y, o)$
    if $fails \geq f$ **then**
      return y
    end if
  end if
  $\textsc{EnforceBudget}(\texttt{rcap})$ ▷ stop round if over cap
end for
return y

Pass-streak acceptance ($p = 3$). To reduce the chance of a one-off false pass from the verifier, we accept a solution only after *three consecutive* successful verifications (pass streak $p = 3$). This simple guard is adopted from the IMO25 solver pipeline [3, 11].

3.2 Budgeting and `rcap`

In practice, iterative loops can run away in cost. Mini Artichokes therefore supports a simple per-round call cap (`rcap`). In our cost-constrained experiments we often use `rcap35`, which stops a refinement round once it consumes roughly 35 API calls (implementation details in our code release [5]). We emphasize that `rcap` is a *budget knob* rather than a conceptual dependency: the core overlap mechanism is unchanged.

3.3 Algorithm

3.4 System diagram

Figure 1 provides a high-level view of the API-only verification-and-refinement loop.

4 Implementation and Attribution

Codebase. Our implementation is available at `https://github.com/pineapplesour/mini-artichokes` [5]. It is a lightweight runner that orchestrates the roles above via prompt templates and API calls.

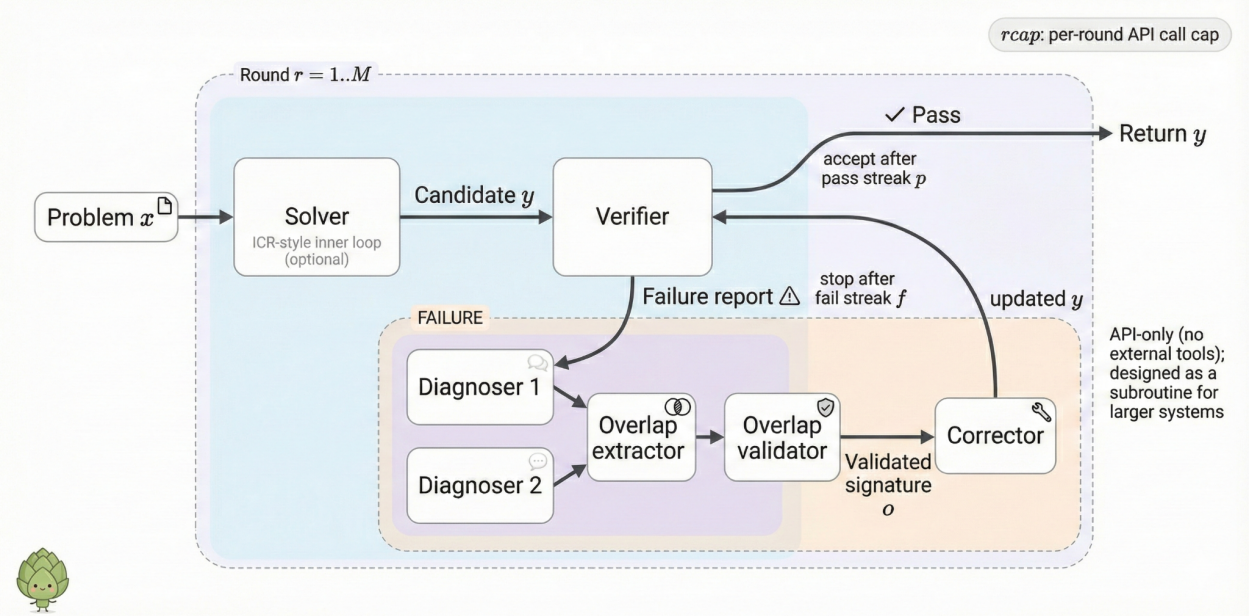

Figure 1: System overview of Mini Artichokes. The acceptance and stopping rules use a pass streak p and a fail streak f, and each round can be bounded by a per-round call cap (`rcap`). Diagram generated with NanoBanana Pro.

We intentionally avoid tool integrations in the reported experiments.

Figure generation. The system overview diagram (Figure 1) was generated with NanoBanana Pro.

Attribution and licensing. Mini Artichokes reuses and adapts ideas (and some scaffolding code) from the open-source repositories [6, 11] and the associated paper [3]. Upstream repositories are permissively licensed (MIT/Apache-2.0 at the time of writing); we cite them explicitly and preserve attribution in our code.

5 Development notes: what we tried and rejected

The code history contains many intermediate variants and ablations. In a paper, it is rarely useful to enumerate every checkpoint name; instead we summarize the most salient *classes* of ideas we tested and why we did *not* keep them in the default Mini Artichokes configuration:

- **Over-aggressive consensus.** Requiring agreement from three or more independent critiques improved precision but was disproportionately expensive and often reduced exploration on hard items.
- **Unvalidated memory growth.** Storing long running "memories" of critiques without validation caused drift (the system starts optimizing to its own folklore). Mini Artichokes keeps feedback short and validated.
- **Checkpoint selection by a learned grader.** A separate grader can select "best" intermediate solutions, but we observed sensitivity to grader prompt phrasing and occasional preference for

superficially plausible but incorrect drafts.

- **Complex routing heuristics.** Routing between multiple sub-pipelines (reading vs. math vs. logic) sometimes helped but introduced brittle failure modes and confounds interpretation. We therefore report the simplest robust loop.

These negative results are not claims of impossibility; they are practical engineering observations from cost-limited experiments.

Beyond the themes above, we explicitly implemented and spot-checked a range of widely used test-time improvement patterns—including iterative self-critique/refinement [4, 7], sampling ensembles (self-consistency) [10], and tree-search style branching (Tree-of-Thoughts) [12]—as well as direct ports of the IMO25 pipeline [3, 11] and ICR-style inner loops [6]. For transparency and to support ablation-style reasoning (e.g., what happens when overlap validation is removed), we provide a compact catalog of these checkpoints and representative spot-check results in Appendix C.

6 Experiments

6.1 Models

For cost reasons, most experiments use a small instruction-tuned model (`gemma-3-27b-it`). We additionally ran a small LEET subset with `gemini-3-flash-preview` to test whether the loop's benefits persist on a different (and typically stronger) commercial API model. The goal is not a vendor comparison but a sanity check on portability.

6.2 Benchmarks and identifiers

We evaluate on the following suites; identifiers in logs match the problem numbering of each source:

- **Putnam 2024** [8, 9]: problems `A1-A6` and `B1-B6`.
- **AIME 2026 (AIME I)** [1]: problems `1-15`.
- **Korean National Civil Service Open Competitive Examination (Grade 5), PSAT—Linguistic Logic** [2]: 40 multiple-choice items from the 2025-03-08 administration (logged as `1-40`).
- **Legal Education Eligibility Test (LEET) 2026, Analytical Reasoning (추리논증)** [13]: a small subset of items (logged as `2,6,13,23,24`).

6.3 Evaluation protocol

For each item we run repeated independent trials (typically 20; only 3 for the LEET subset due to cost). We report:

1. **Full-suite accuracy** (Appendix tables).
2. **Hard-item lift**: to avoid the "easy items dominate" effect, we also report performance on the subset of items that pass@1 solves *at most once* in 20 trials (for PSAT), which makes differences visible without changing the underlying data.

Table 1: Selected hard-item gains (Mini Artichokes vs. pass@1). Cells are Correct/Attempts over repeated trials; Δ is Mini minus pass@1.

Benchmark	**QID**	**pass@1**	**Mini Artichokes**	**Δ**
PSAT 2025 (Linguistic Logic)	38	0/20	20/20	+20
PSAT 2025 (Linguistic Logic)	28	1/20	20/20	+19
PSAT 2025 (Linguistic Logic)	3	0/20	17/20	+17
PSAT 2025 (Linguistic Logic)	14	0/20	11/19	+11
PSAT 2025 (Linguistic Logic)	34	0/20	8/20	+8
PSAT 2025 (Linguistic Logic)	31	0/20	7/20	+7
LEET 2026 (AR subset)	2	0/3	2/3	+2
LEET 2026 (AR subset)	24	0/3	1/3	+1
Putnam 2024	B2	0/20	10/20	+10
Putnam 2024	A4	0/20	7/20	+7
Putnam 2024	A2	0/20	6/20	+6
AIME 2026	P5	0/20	19/20	+19

6.4 Results

Table 1 highlights representative *hard* items where pass@1 rarely succeeds but Mini Artichokes succeeds repeatedly. We emphasize that these rows are *selected for difficulty* (explicit criterion above); full per-item tables are provided in Appendix A, including cases where Mini Artichokes regresses.

PSAT hard subset. On PSAT Linguistic Logic, the overall suite accuracy improves modestly (58.4% → 61.5%), but the effect concentrates on the hardest questions. Under a best-of-20 scoring view (each PSAT item counts if it is answered correctly at least once over 20 trials; 2.5 points per item), pass@1 reaches 62.5/100 (25/40 items) while Mini Artichokes reaches 92.5/100 (37/40 items), a +30.0-point gap. This "at-least-once" view makes the intended behavior explicit: Mini Artichokes is designed to turn pass@1's *zero-success* instances into *some* success, rather than to dominate on already-easy items. Concretely, across the 40 PSAT items, 25 are solved at least once by both methods, 12 are solved at least once only by Mini Artichokes, and 3 are never solved by either; in this run we do not observe any item that pass@1 solves at least once while Mini Artichokes has zero successes. Occasionally Mini Artichokes can be less accurate than pass@1 on items that pass@1 already solves reliably, which we suspect is due to over-skepticism ("overthinking") in the critique loop; we expect such regressions to diminish as base models strengthen, but leave a systematic study to future work.

LEET subset selection. We did not evaluate the full LEET exam due to cost. Most LEET items we spot-checked are solved by both methods; to make the comparison informative, we report the two hardest tested items (QID 2 and 24) where pass@1 failed in all trials but Mini Artichokes succeeded at least once (Table 1). Full subset results are in Appendix.

6.5 Cost

Mini Artichokes is not free: it spends additional calls and tokens. On PSAT, the average call count is 26.6 per trial (vs. 1 for pass@1). We view this as a controlled trade-off rather than an attempt to

"beat" strong tool-augmented systems; the architecture is designed to be a small module that can later be combined with tools, retrieval, or external validation, where the downstream gains may be larger.

7 Limitations and Outlook

Our experiments are limited in breadth (especially LEET) and are constrained to a tool-free regime. Mini Artichokes should not be interpreted as a replacement for external verification; rather, it is a conservative internal refinement primitive. We expect the same loop to benefit from stronger base models and from integration into larger systems that provide real feedback signals (tests, formal checkers, retrieval), but we leave those extensions to future work.

A Full per-item tables

This appendix provides complete per-item comparisons for transparency, including items where Mini Artichokes performs worse than pass@1. These tables correspond to the raw "Problem summary" blocks in the attached experiment log.

A.1 PSAT 2025 Linguistic Logic

Table 2: Per-item results on PSAT 2025 Linguistic Logic (40 items). Each cell is Correct/Attempts over repeated trials; Δ is Mini minus pass@1; Fold is the accuracy ratio (Mini/pass@1), reported as ∞ when pass@1 is 0 but Mini > 0, and N/A when both are 0.

QID	pass@1	Mini Artichokes	Δ	Fold
1	20/20	20/20	+0	1×
2	20/20	18/20	-2	0.90×
3	0/20	17/20	+17	∞
4	20/20	20/20	+0	1×
5	20/20	20/20	+0	1×
6	20/20	12/20	-8	0.60×
7	0/20	3/20	+3	∞
8	19/20	20/20	+1	1.05×
9	20/20	20/20	+0	1×
10	20/20	20/20	+0	1×
11	11/20	9/20	-2	0.82×
12	0/20	0/20	+0	N/A
13	20/20	3/20	-17	0.15×
14	0/20	11/19	+11	∞
15	0/20	3/20	+3	∞
16	0/20	3/20	+3	∞
17	20/20	7/20	-13	0.35×
18	20/20	10/20	-10	0.50×
19	20/20	20/20	+0	1×

QID	pass@1	Mini Artichokes	Δ	Fold
20	20/20	6/20	-14	0.30×
21	20/20	20/20	+0	1×
22	0/20	4/20	+4	∞
23	20/20	18/20	-2	0.90×
24	20/20	20/20	+0	1×
25	20/20	20/20	+0	1×
26	20/20	20/20	+0	1×
27	16/20	20/20	+4	1.25×
28	1/20	20/20	+19	20×
29	20/20	20/20	+0	1×
30	0/20	1/20	+1	∞
31	0/20	7/20	+7	∞
32	0/20	4/20	+4	∞
33	20/20	13/20	-7	0.65×
34	0/20	8/20	+8	∞
35	20/20	12/20	-8	0.60×
36	0/20	2/20	+2	∞
37	0/20	0/20	+0	N/A
38	0/20	20/20	+20	∞
39	20/20	20/20	+0	1×
40	0/20	0/20	+0	N/A

A.2 LEET 2026 Analytical Reasoning subset

Table 3: Per-item results on a small LEET 2026 Analytical Reasoning (추리논증) subset (5 items). Cells are Correct/Attempts; Δ is Mini minus pass@1; Fold is the accuracy ratio (Mini/pass@1), reported as ∞ when pass@1 is 0 but Mini > 0, and N/A when both are 0.

QID	pass@1	Mini Artichokes	Δ	Fold
2	0/3	2/3	+2	∞
6	0/3	0/3	+0	N/A
13	3/3	2/3	-1	0.67×
23	3/3	2/3	-1	0.67×
24	0/3	1/3	+1	∞

A.3 Putnam 2024

Table 4: Per-problem results on Putnam 2024 (A1–A6, B1–B6). Cells are Correct/Attempts; Δ is Mini minus pass@1; Fold is the accuracy ratio (Mini/pass@1), reported as ∞ when pass@1 is 0 but Mini > 0, and N/A when both are 0.

QID	pass@1	Mini Artichokes	Δ	Fold
A1	20/20	16/20	-4	0.80×
A2	0/20	6/20	+6	∞
A3	20/20	18/20	-2	0.90×
A4	0/20	7/20	+7	∞
A5	0/20	1/20	+1	∞
A6	4/20	0/20	-4	0×
B1	1/20	5/20	+4	5×
B2	0/20	10/20	+10	∞
B3	20/20	1/20	-19	0.05×
B4	0/20	2/20	+2	∞
B5	0/20	4/20	+4	∞
B6	0/20	1/20	+1	∞

A.4 AIME 2026

Table 5: Per-item results on AIME 2026 (Problems 1–15). Cells are Correct/Attempts; Δ is Mini minus pass@1; Fold is the accuracy ratio (Mini/pass@1), reported as ∞ when pass@1 is 0 but Mini > 0, and N/A when both are 0.

QID	pass@1	Mini Artichokes	Δ	Fold
P1	20/20	19/20	-1	0.95×
P2	0/20	1/20	+1	∞
P3	18/20	4/20	-14	0.22×
P4	0/20	0/20	+0	N/A
P5	0/20	19/20	+19	∞
P6	20/20	20/20	+0	1×
P7	0/20	1/20	+1	∞
P8	2/20	10/20	+8	5×
P9	0/20	2/20	+2	∞
P10	0/20	0/20	+0	N/A
P11	0/20	0/20	+0	N/A
P12	1/20	6/20	+5	6×
P13	0/20	0/20	+0	N/A
P14	0/20	0/20	+0	N/A
P15	0/20	0/20	+0	N/A

B Suite-level summary

Table 6: Suite-level accuracy and average calls per trial (for context).

Suite	pass@1	Mini Artichokes	pass@1 calls	Mini calls
AIME 2026 (15 items, 20 trials each)	61/300 (20.3%)	82/300 (27.3%)	1.0	67.28
PSAT 2025 Linguistic Logic (40 items, ~20 trials each)	467/800 (58.4%)	491/799 (61.5%)	1.0	26.55
Putnam 2024 (12 problems, 20 trials each)	65/240 (27.1%)	71/240 (29.6%)	1.0	121.07
LEET 2026 Analytical Reasoning (5 items, 3 trials each)	6/15 (40.0%)	7/15 (46.7%)	1.0	24.40

C Additional architecture variants and ablations

Mini Artichokes was developed by iterating over many internal "checkpoint" architectures in the released runner code. The main paper reports only the final configuration (and pass@1) because a full baseline zoo would balloon cost and space; however, to support the claim that we tested a broad menu of plausible alternatives (and to enable ablation-style reasoning), we summarize a compact catalog and a few representative spot-check results here.

ID convention. We refer to checkpoints by their internal runner IDs (e.g., `test37`); these are fixed prompt+budget configurations in the released code (not different base models), and we keep the IDs so the tables match the development logs exactly.

C.1 Checkpoint families (with representative IDs)

The following families were implemented and exercised during development:

- **IMO25-style verifier loop** [3, 11]: direct ports of the IMO25 verification-and-refinement pipeline (e.g., `test20-test21`).
- **ICR-style inner loops** [6]: multiple ICR modes (React/Deepthink/Adaptive/Agentic/Contextual; e.g., `test33-test38`).
- **Iterative critique–revise loops** [4, 7]: alternating critique and correction loops with different verifier prompts/budgets (e.g., `test31`, `test40-test43`).
- **Verification checklist / question decomposition ("System 1.2")**: checklist-based verifiers with memory and routing heuristics (`system-1.2`) and feature ablations (e.g., `test1`: baseline step-logic Red Team; `test2`: methodology vote/selection; `test3`: final "trial" verification; `test4-test5`: checklist-criteria generation; `test12-test15`: counterexample post-passes; `test16-test17`: persona round-robin).
- **Sampling ensembles (self-consistency)** [10]: multi-solver sampling + best-of/selection variants (e.g., `test6`).
- **Branching search (Tree-of-Thoughts)** [12]: explicit multi-branch exploration with prune/regrow (e.g., `test8-test9`).

Table 7: Development sweep on a hard Korean reading-comprehension item (2025-03 national mock exam, Korean Q12). Attempts can differ across checkpoints when runs exceed a fixed budget or are terminated early.

Checkpoint	Correct/Attempts	Accuracy	Avg calls	Avg tokens
`test37`	10/10	100%	10	27,760
`test33`	8/10	80%	7	18,000
`test34`	8/10	80%	12	33,007
`test31`	4/6	66.67%	198.17	513,030
`test43`	13/20	65.0%	23.95	62,103
`test43-2`	13/20	65.0%	22.60	45,840
`test21`	5/10	50%	19	36,029
`test22`	5/10	50%	115.30	269,588
`test35`	3/10	30%	7	19,248
`test11`	2/10	20%	8.20	19,696
`test38`	1/10	10%	4	11,803
`pass@1`	0/10	0%	1	1,508
`test1`	0/10	0%	6.70	13,295
`test4`	0/10	0%	7	18,666
`test5`	0/10	0%	21.90	52,366
`test6`	0/10	0%	32.50	73,263
`test7`	0/1	0%	32	83,318
`test8`	0/5	0%	65.80	145,956
`test9`	0/3	0%	132.67	344,409
`test12`	0/1	0%	60	165,512
`test17`	0/2	0%	89	238,092
`test32`	0/10	0%	7	16,272
`test36`	0/10	0%	6	11,064

C.2 Broad sweep on one reading-comprehension item

Table 7 shows a broad sweep of representative checkpoints on a single hard Korean reading-comprehension item (2025-03 high-school national mock exam, Korean Q12; ⑤ is correct). This is a development sanity check: it is *not* part of the main benchmark suites, and should not be over-interpreted as a benchmark claim.

Although `test37` tops this single reading-comprehension item, it was not the most robust choice across the PSAT-style logical-reasoning items that motivated this work. In particular, `test37` can fail catastrophically on some Linguistic Logic items (see Table 8, item 17), while the critique–revise family around `test43` retains non-trivial accuracy on the same hard cases. This motivated using `test43` as the base loop, and then selecting `test43-2` (Mini Artichokes) via feature ablations below.

C.3 PSAT/logic spot-checks

Table 8 compares four representative checkpoints on a handful of PSAT-style items (10 trials each) from the same development log: (i) pass@1, (ii) an IMO25-style port (`test21`), (iii) an ICR contextual mode (`test37`), and (iv) an early Mini-Artichokes-like critique–revise loop (`test43`). These spot-checks help explain the selection path: `test37` can be excellent on some items (including

Table 8: PSAT/logic spot-checks from the development log (10 trials each). Entries are Correct/Attempts with Avg calls in parentheses. SJ = Situation Judgment; LL = Linguistic Logic.

Item	pass@1	test21	test37	test43
9 (SJ)	6/10 (1.4)	5/10 (12.6)	7/10 (10.0)	9/10 (19.4)
33 (LL)	8/10 (1.0)	4/10 (12.9)	9/10 (10.0)	7/10 (23.0)
17 (LL)	0/10 (1.0)	7/10 (12.3)	0/10 (10.0)	5/10 (18.0)
34 (LL)	0/10 (1.0)	5/10 (12.7)	–	4/10 (19.4)
14 (LL)	0/10 (1.0)	0/10 (29.4)	–	3/10 (24.3)

Table 9: Mini Artichokes feature ablations on PSAT Linguistic Logic item 17 (10 trials).

Checkpoint	Features	Correct/Attempts	Accuracy	Avg calls	Avg tokens
test43-2	O	8/10	80.0%	18.00	28,120
test43-2-3	O+M	7/10	70.0%	36.00	55,321
test43-1	G	6/10	60.0%	37.50	60,237
test43-3	M	6/10	60.0%	36.00	55,405
test43-1-2	G+O	4/10	40.0%	36.00	55,338
test43-1-2-3	G+O+M	4/10	40.0%	36.00	55,459
test43-1-3	G+M	3/10	30.0%	36.00	59,847

Table 7 and LL item 33), but brittle on others (e.g., 0/10 on LL item 17), whereas `test43` stays consistently non-zero on the hardest LL items where pass@1 is 0/10. These runs are separate from the full-suite PSAT evaluation in Appendix A.

C.4 Mini Artichokes feature ablations

We used three feature blocks that can be toggled in development checkpoints: (G) graded checkpoint selection (suffix `-1`), (O) overlap diagnosis + flaw validation (suffix `-2`, the core Mini Artichokes mechanism), and (M) outlier insight memory (suffix `-3`). Empirically, (O) was the most consistently beneficial toggle in our development log, which is why the main paper reports `test43-2` as the default configuration. Tables 9–10 show ablations on two PSAT Linguistic Logic items.

References

[1] *2026 AIME I Problems (AoPS Contest Collections PDF)*. Art of Problem Solving Wiki. 2026. URL: https://artofproblemsolving.com/wiki/index.php/2026_AIME_I_Problems (visited on 02/15/2026).

[2] *Cyber State Examination Center: Multiple-choice exam problems and answer keys (선택형 문제/정답)*. Navigate to 5급 공개경쟁채용시험(행정) to access PSAT problems and answer keys. Ministry of Personnel Management (Republic of Korea). URL: https://www.gosi.kr/cop/bbs/gosiQnaChoiceExam.do (visited on 02/15/2026).

[3] Yichen Huang and Lin F. Yang. "Winning Gold at IMO 2025 with a Model-Agnostic Verification-and-Refinement Pipeline". In: (2025). arXiv preprint. arXiv: 2507.15855 [cs.AI]. URL: https://arxiv.org/abs/2507.15855.

Table 10: Mini Artichokes feature ablations on PSAT Linguistic Logic item 14 (attempts differ when runs exceed budget).

Checkpoint	Features	Correct/Attempts	Accuracy	Avg calls	Avg tokens
test43-2	O	9/9	100.0%	30.67	73,132
test43-1	G	5/10	50.0%	47.40	96,203
test43-2-3	O+M	5/10	50.0%	45.20	92,502
test43-1-2	G+O	4/10	40.0%	46.90	84,047
test43-1-3	G+M	3/9	33.3%	56.11	115,370
test43-3	M	3/10	30.0%	38.60	70,099
test43-1-2-3	G+O+M	1/10	10.0%	46.00	87,722

[4] Aman Madaan et al. *Self-Refine: Iterative Refinement with Self-Feedback*. 2023. arXiv: 2303.17651 [cs.CL]. URL: https://arxiv.org/abs/2303.17651.

[5] Hyeongbin Park. *Mini Artichokes*. Code release (GitHub repository). URL: https://github.com/pineapplesour/mini-artichokes (visited on 02/15/2026).

[6] ryoiki-tokuiten. *Iterative-Contextual-Refinements*. GitHub repository. Licensed under Apache-2.0. URL: https://github.com/ryoiki-tokuiten/Iterative-Contextual-Refinements (visited on 02/15/2026).

[7] Noah Shinn et al. *Reflexion: Language Agents with Verbal Reinforcement Learning*. 2023. arXiv: 2303.11366 [cs.AI]. URL: https://arxiv.org/abs/2303.11366.

[8] *The 85th William Lowell Putnam Mathematical Competition (2024): Problems for Session A*. Mathematical Association of America. 2024. URL: https://maa.org/wp-content/uploads/2025/02/2024Putnam_A-1.pdf (visited on 02/15/2026).

[9] *The 85th William Lowell Putnam Mathematical Competition (2024): Problems for Session B*. Mathematical Association of America. 2024. URL: https://maa.org/wp-content/uploads/2025/02/2024Putnam_B-1.pdf (visited on 02/15/2026).

[10] Xuezhi Wang et al. *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. 2022. arXiv: 2203.11171 [cs.CL]. URL: https://arxiv.org/abs/2203.11171.

[11] Lin Yang and Yichen Huang. *IMO25: IMO 2025 Problem Solver*. GitHub repository. Licensed under MIT. URL: https://github.com/lyang36/IMO25 (visited on 02/15/2026).

[12] Shunyu Yao et al. *Tree of Thoughts: Deliberate Problem Solving with Large Language Models*. 2023. arXiv: 2305.10601 [cs.CL]. URL: https://arxiv.org/abs/2305.10601.

[13] *법학적성시험 문제 해설: LEET 추리논증 (2026학년도)*. ISBN: 978-89-200-5488-4. 법학전문대학원협의회. 2025. URL: https://press.knou.ac.kr/goods/commonViewUrl.do?condCmdtCode=9788920054884 (visited on 02/15/2026).