Autoreason: Self-Refinement That Knows When to Stop

SHL0MS | HERMES AGENT
github.com/NousResearch/autoreason

Abstract. Iterative self-refinement fails for three reasons: *prompt bias*, where adversarial critique prompts cause models to hallucinate problems that don't exist; *scope creep*, where each revision pass expands the document's scope unchecked; and *lack of restraint*: models almost never decline to make changes, even when the output is already good. Together these guarantee progressive degradation. We present autoreason, which addresses all three by structuring each iteration as a three-way choice: the unchanged incumbent (A), an adversarial revision (B), and a synthesis (AB). Fresh agents with no prompt history or session context judge the candidates via blind Borda count, ensuring that "do nothing" is always a first-class option, and the evaluators share none of the biases that produced the revisions. The method's value is greatest at mid-tier models, where the gap between generation capability and self-evaluation capability is widest. With Haiku 3.5 ($\sim$10$\times$ cheaper than Sonnet 4), autoreason scored 42/42 Borda across three tasks, a perfect sweep, while standard refinement baselines *degraded* the same model's outputs below its unrefined single pass. Across five model tiers (Llama 8B, Gemini Flash, Haiku 3.5, Haiku 4.5, Sonnet 4), the advantage peaks at mid-tier and diminishes at both extremes: models too weak to generate diverse alternatives (Llama) or too strong to need external evaluation (Sonnet 4.6). Haiku 4.5 confirms the transition: at 60% private-test accuracy on code (matching Sonnet single-pass), autoreason's gains on held-out tests vanish entirely, despite a 4pp lift on visible tests.

We validate across 5 subjective writing tasks, 150 competitive programming problems (CodeContests) at four model tiers, a five-tier model capability comparison, and systematic ablations of judge count, aggregation method, component necessity, and length bias. In code, autoreason achieves 77% private-test accuracy vs. 73% for single-pass with Sonnet 4.6. With weaker Haiku 3.5 authors, autoreason outperforms best-of-6 sampling (40% vs. 31%) at matched compute, while critique-and-revise degrades below single-pass. With stronger Haiku 4.5 authors, the advantage disappears on held-out tests (60% for all strategies), confirming the method's value is bounded by the generation-evaluation gap. Length-controlled evaluation confirms gains survive when outputs are truncated to matched word counts. The practical recommendation: autoreason operates in the space where models can generate diverse alternatives but cannot reliably choose between them.

---

1 Introduction

Iterative LLM self-refinement is widely assumed to improve outputs, but the evidence is mixed. The failure modes are structural, not incidental. First, *prompt bias*: a critique prompt like "find problems with this document" will almost always produce criticism, even when the document is good. The model hallucinates flaws to satisfy the instruction, a form of sycophantic behavior where the model prioritizes adherence to the prompt over accurate assessment [16, 17]. Second, *scope creep*: each revision pass adds detail, restructures, and expands, because both models and judges exhibit systematic verbosity bias; longer outputs are preferred regardless of quality [18, 19]. Third, *lack of restraint*: models rarely decline to make changes, even when told to be conservative.

The instruction to revise overrides any hedging, and self-correction without external feedback is largely ineffective [5, 20]. Self-Refine [4] demonstrates that critique-and-revise loops can help but provides no mechanism to detect when revisions make things worse. SlopCodeBench [2] documents progressive degradation in iterative code generation, and Shukla et al. [23] show the same pattern for security properties in iterative code refinement. The question is not whether self-refinement can work, but *when* it works and what makes it fail.

We present autoreason, extending Karpathy's autoresearch [1] beyond objective metrics to domains where quality is subjective or multi-dimensional. The key insight is that "do nothing" must be a first-class option. Each iteration produces three competing versions: the unchanged incumbent (A), an adversarial revision (B), and a synthesis (AB), evaluated by a blind panel of fresh agents that share none of the prompt biases or session context that produced the candidates. The method constructs a fitness function through the evaluation process itself, and the incumbent survives by default if the alternatives aren't genuinely better. Component ablations confirm that both the adversarial revision and the synthesis are necessary: removing either collapses the tournament, and the incumbent wins immediately without meaningful competition. Length-controlled evaluation confirms the quality gains are not a verbosity artifact.

Our central finding is that **the value of structured refinement depends on the gap between a model's generation capability and its self-evaluation capability**. Self-refinement fails because evaluation is harder than generation [5]. Autoreason bridges this gap with external evaluation. Across five model tiers, the advantage peaks at mid-tier models where the gap is widest: Haiku 3.5 achieved a perfect 42/42 Borda while standard baselines *degraded* its outputs below the unrefined starting point (Section 7). At the bottom (Llama 8B), the model is too weak to generate diverse alternatives for the tournament. At the top (Sonnet 4.6), the gap is narrow enough that simple methods suffice (Section 5). Haiku 4.5 marks the transition: strong enough that autoreason's private-test gains vanish on code, despite visible-test improvements (Section 6). The method operates in the practical middle, the range where most cost-conscious LLM deployment lives, and where the generation-evaluation gap is large enough to exploit.

Our experiments identify four conditions for the method to succeed: (1) **external verification**: providing evaluation capability the generating model lacks; (2) **constrained scope**: bounded improvement space; (3) **structured reasoning**: explicit failure analysis rather than reactive editing; and (4) **sufficient decision space**: the task admits genuinely different valid approaches. We test across subjective writing (5 open-ended tasks, 3 constrained tasks, 8 remedy experiments), competitive programming (150 CodeContests problems across four model tiers), a five-tier model capability comparison (Llama 8B through Sonnet 4.6), and systematic ablations of the method's components (judge count, aggregation, synthesis vs. revision, length bias). These conditions are consistent with our results but should be treated as hypotheses supported by evidence, not established laws.

2 Method

Critic. A fresh agent identifies problems in the current incumbent A. Finds problems only; no fixes. Optionally receives ground-truth data for fact-checking (Section 8).

Author B. A fresh agent revises A based on the critique. Sees the task, A, and the critique. No

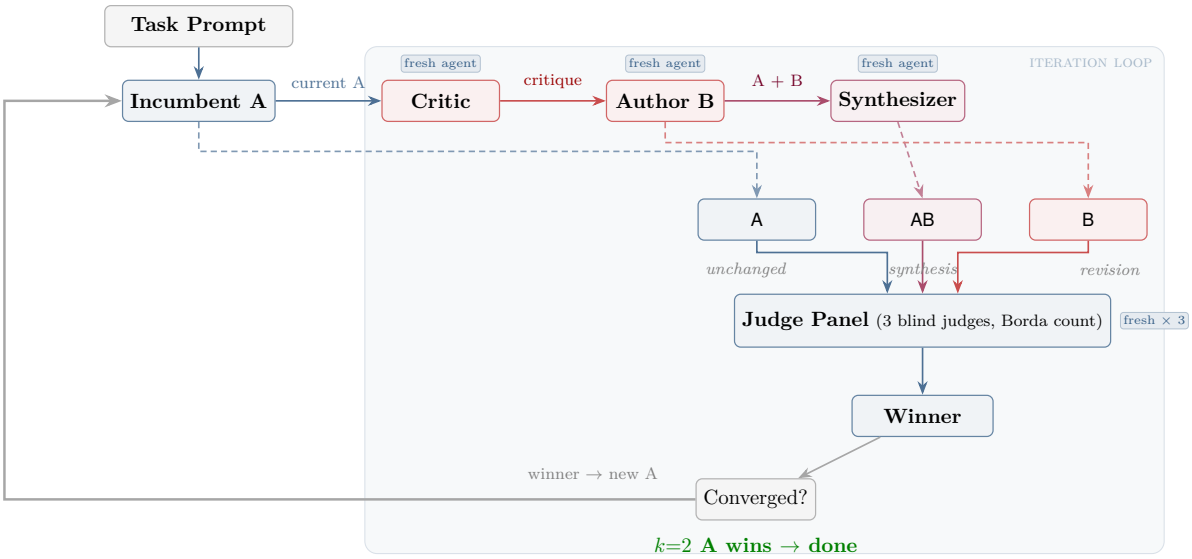

Figure 1. The autoreason loop. Each pass produces three candidates: the unchanged incumbent (A), a synthesis (AB), and an adversarial revision (B). A blind judge panel selects the winner via Borda count. If A survives k=2 consecutive passes, the loop terminates. Every role is a fresh, isolated agent.

drafting history.

Synthesizer. A fresh agent receives A and B with randomized labels. Produces AB by taking the strongest elements of each.

Judge Panel. 3 fresh agents rank A, AB, B via Borda count (3/2/1 points). Randomized labels and presentation order. Conservative tiebreak: incumbent wins ties. Convergence at k=2 consecutive A wins. We use 3 judges in-loop to balance per-pass cost against evaluation stability, and 7 judges for final blind comparisons to increase statistical power where cost is amortized over a single evaluation.

Every role is a fresh, isolated agent with no shared context. The original task prompt anchors all evaluation. We chose k=2 based on a sensitivity analysis across 17 trajectories (Table 23). k=1 terminates prematurely (94% of runs saw subsequent displacement). k=2 converges 100% at an average of 3.9 passes. k=3 fails to converge on 24% of trajectories at 2× the cost, and k=4 fails on 47%. While runs that continue past k=2 convergence do see further displacement, this does not improve quality: a 7-judge panel chose the k=2 output over the k=4 output by 6–1 (Section 4.5), and Elo ratings plateau by pass 5–10 regardless. Temperature was set to 0.8 for authors (encouraging diverse revisions) and 0.3 for judges (encouraging consistent evaluation); these values were chosen based on preliminary runs and not formally ablated. Exact prompts are provided in Appendix A.

2.1 Formal Framing

Let d_t denote the incumbent document at pass t. Each pass produces candidates $\{d_t, B(d_t), S(d_t, B(d_t))\}$ where B is the adversarial revision operator and S is the synthesis operator. A judge panel J defines a stochastic preference relation: $J(x, y) \in [0, 1]$ is the probability that x is ranked above y. The winner is selected by Borda aggregation over n judges:

$$d_{t+1} = \arg \max_{c \in \{A,B,AB\}} \sum_{i=1}^{n} (3 - r_i(c))$$

where $r_i(c)$ is judge i's rank for candidate c, with ties broken in favor of d_t.

This defines an iterative process over document states. Convergence ($d_{t+1} = d_t$ for k consecutive passes) occurs when the incumbent survives repeated challenges. Using multiple judges provides some robustness against individual judge errors, though same-model judges share systematic biases that limit the effective diversity of the panel.

There is a loose analogy to PSRO [11], where each pass adds a candidate to an empirical game, but autoreason lacks PSRO's meta-strategy computation, Nash equilibrium search, or formal game matrix. Similarly, while the Condorcet jury theorem suggests panel accuracy should increase with n for independent judges, our same-model judges violate the independence assumption. We do not claim game-theoretic or voting-theoretic guarantees. Our retroactive analysis (Section 4.5) examines the empirical properties of the preference structure: preferences are near-transitive across 91 passes (1.1% Condorcet cycle rate), and Elo ratings plateau early. These are descriptive observations, not proofs of optimality.

3 Experiments

All experiments used claude-sonnet-4-20250514 (temp=0.8 authors, 0.3 judges, max_tokens=4096) unless otherwise noted. Five tasks: (1) go-to-market strategy, (2) notification system design, (3) remote work policy, (4) competitive positioning, (5) incident response. Each task ran autoreason to convergence (k=2) plus four baselines at 15 passes each from the same initial output, evaluated by a 7-judge blind panel. The paper-writing experiment (Section 8) used claude-opus-4. Scaling experiments (Section 5) used claude-sonnet-4-6-20250514 (hereafter "Sonnet 4.6").

Baselines. We compared against four iterative prompting strategies, each applied for 15 passes from the same initial document: (1) *Conservative*: "Make minimal improvements while preserving what works"; (2) *Improve this*: "Improve this document" with no further guidance; (3) *Harsh critic*: "Critically evaluate and rewrite, fixing all weaknesses"; (4) *Critique & revise*: a two-step loop where the model first produces a structured critique, then revises the document to address it (closest to Self-Refine [4]). All baselines use a single agent per pass with no judge panel or adversarial structure. Baseline prompts are provided in Appendix A.

Compute budget. Autoreason requires ∼6 LLM calls per pass (critic, author B, synthesizer, 3 judges) and typically runs 10–25 passes to convergence (∼60–150 calls per task), while each baseline requires ∼1 call per pass (∼15 calls total). This roughly 6× per-pass cost (and larger total cost due to more passes) is a real tradeoff; autoreason trades compute for evaluation quality.

4 Results

4.1 Single Pass vs. Iteration

Method	Calls	T1	T2	T3	T4	T5	Avg
Harsh critic	1	**30**	**31**	23	**30**	**31**	**29.0**
Critique & rev.	1	**30**	30	**31**	28	23	28.4
Autoreason	5	24	22	24	17	24	22.2
Improve this	1	14	14	18	23	20	17.8
Conservative	1	7	8	9	7	7	7.6

Table 1. Single-pass Borda scores (max 35). Bold indicates task winner. Autoreason places 3rd at 5× the cost.

Method	T1	T2	T3	T4	T5	Avg
Autoreason	**35**	19	**29**	25	**31**	**27.8**
Critique & rev.	13	**22**	26	**32**	19	22.4
Harsh critic	18	19	27	20	26	22.0
Conservative	21	19	15	18	15	17.6
Improve this	18	11	8	10	14	12.2

Table 2. Iterative Borda scores (max 35) at 15 passes. Autoreason wins tasks 1, 3, 5; critique-and-revise wins tasks 2, 4.

Single-pass autoreason averaged 22.2 Borda, placing 3rd on 4/5 tasks (Table 1). With iteration, it won 3 of 5 tasks (avg 27.8, rank 1.4), outperforming all single-pass methods (Table 2, Figure 2). The judge panel acts as a filter: revisions that score lower than the incumbent are rejected, while baselines accumulate drift unchecked. This filtering is defined by LLM preference, not an independent quality measure, so its effectiveness is bounded by the judges' biases. The scaling results (Section 5) demonstrate where this protection breaks down.

Autoreason excels on tasks with genuine tradeoffs (strategy, policy, incident response). Critique-and-revise wins on tasks with concrete technical requirements (system design, competitive positioning) where a direct find-and-fix loop is more efficient (Figure 3).

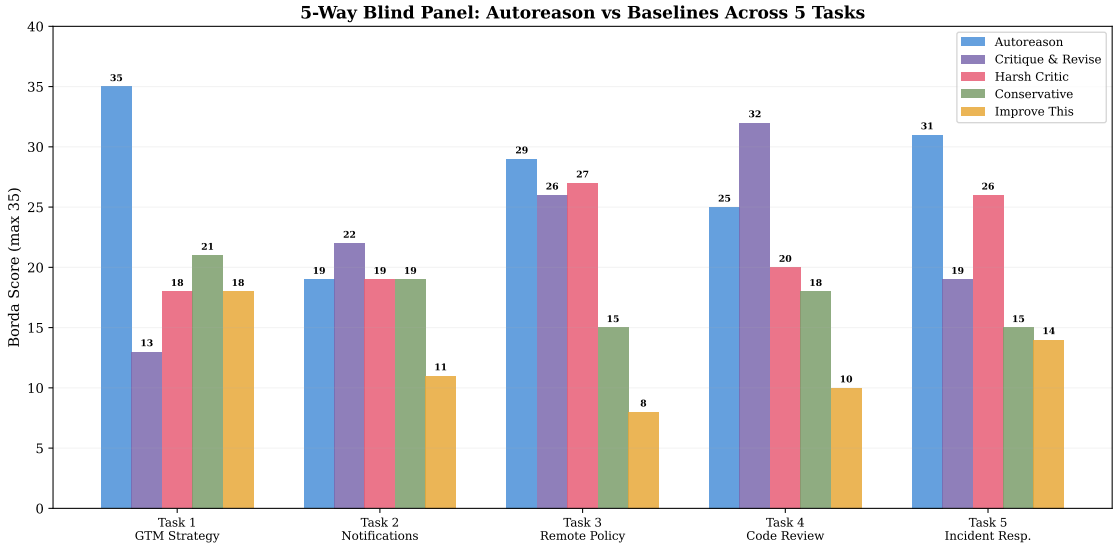

Figure 2. Per-task Borda scores at 15 iterative passes.

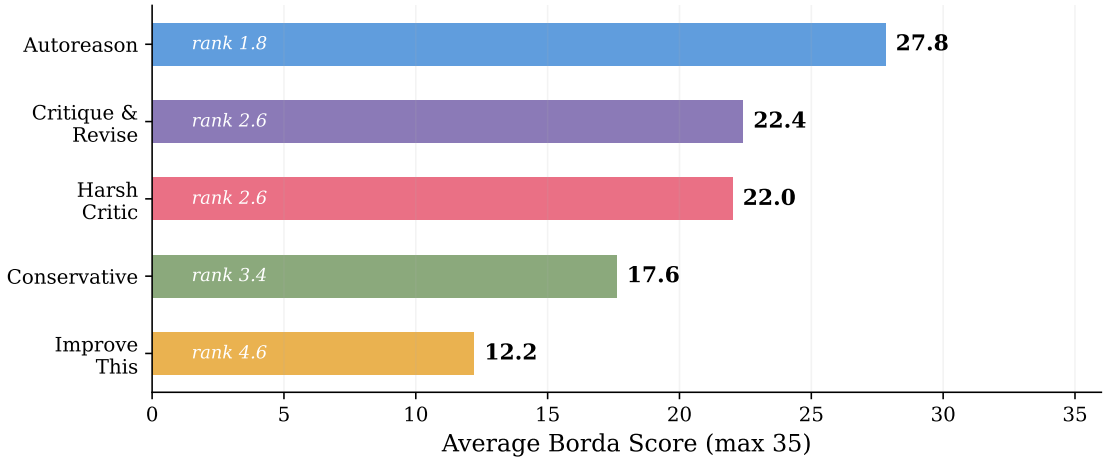

Figure 3. Aggregate performance. Autoreason (27.8 avg Borda, rank 1.4) never placed below 2nd.

4.2 Qualitative Improvement

The initial Task 1 output was a generic startup playbook: $49/user pricing, $100K MRR target with a 3-person team, no customer validation. After 14 passes, the converged output contained quantified pain points, team-based pricing matching the buying motion, customer validation details, competitive positioning against named tools, and unit economics. The adversarial process replaced vague aspirational claims with internally consistent specifics. These details are LLM-generated and not verified against real-world data; the improvement is in internal coherence and specificity, not factual accuracy.

4.3 Convergence and Trajectory

All 5 tasks converged at threshold k=2, requiring 9–28 passes (Figure 4). Policy tasks converge faster (10 passes) than multi-stakeholder operational processes (28 passes).

Figure 5 shows the Task 1 trajectory. At pass 15, A had won twice consecutively. Eleven more passes followed. A 7-judge panel comparing pass 15 vs pass 25 chose pass 15 by 6–1. Post-convergence passes degraded quality in this case.

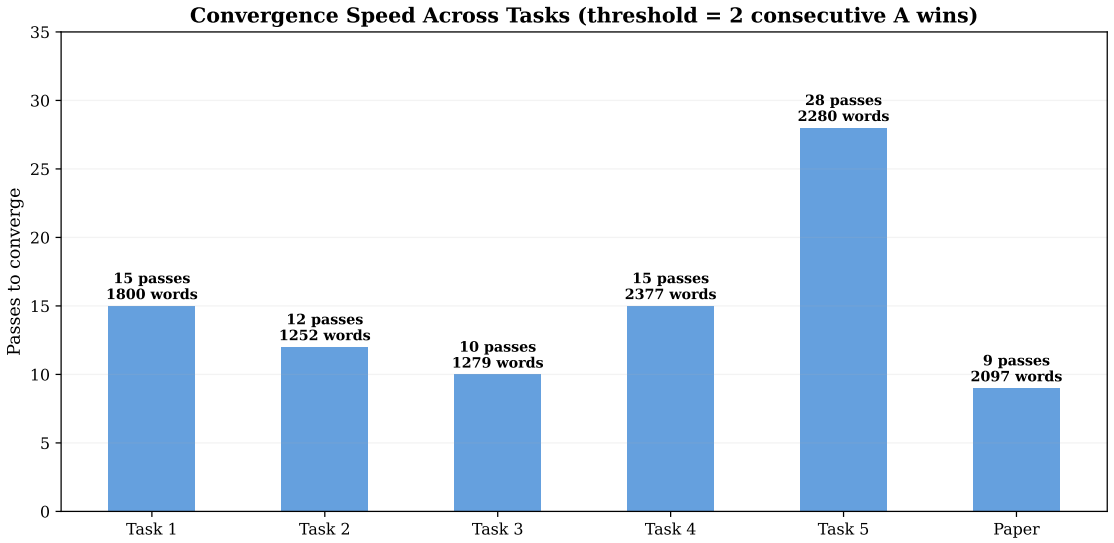

Figure 4. Passes to convergence across all tasks. Task 5 required the most (28); the paper (9) the least.

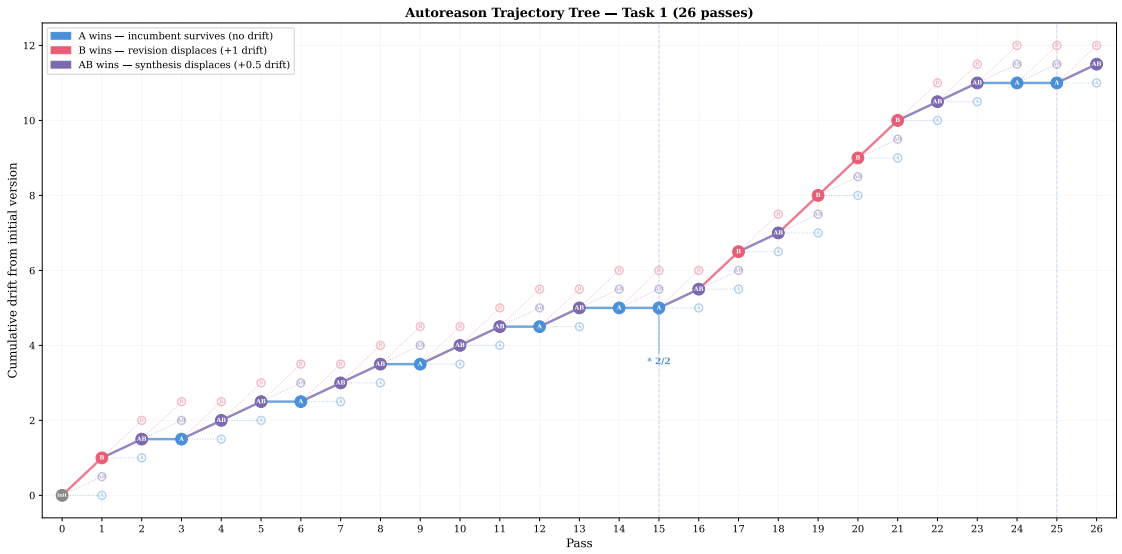

Figure 5. Trajectory tree, Task 1 (26 passes). Stars mark convergence points.

4.4 Chain-of-Thought Judges

Adding "think step by step about each version" to the judge prompt, with no architecture changes, dramatically improves convergence (Table 3). CoT judges produce scores of 8–9 for A wins (vs baseline's 6–7), acting as a debiasing mechanism: judges must articulate specific strengths before committing to a ranking, preventing verbosity bias and pattern-matching.

Judge Variant	Task 1	Task 2
Baseline (holistic)	14–15 passes	12 passes
Chain-of-thought	**5 passes**	**8 passes**
Decomposed (3 specialists)	7 passes	—

Table 3. CoT reduces convergence 3× on Task 1 (1.5× on Task 2). All subsequent experiments use CoT judges.

4.5 Robustness

Monte Carlo (Sonnet 4). Five independent runs of Task 1: 4/5 converged (80%). Convergence pass varied (6, 8, 14, 17) but final word counts clustered tightly (1451–1660), suggesting a consistent quality ceiling regardless of path (Figure 6).

Monte Carlo (Sonnet 4.6, constrained). Five independent runs of the constrained policy task: 0/5 converged (all hit the 25-pass cap). Word counts spread (481–853, std=135 vs. Sonnet 4's 1451–1660, std=83). A 7-judge cross-evaluation of all 5 final outputs showed Borda std=8.3 (spread), compared to the tight clustering expected from a converging system. Sonnet 4.6 produces competitive outputs (the best run scored 31/35 against baselines), but the quality is trajectory-dependent, not deterministic. The stronger model's ability to always generate compelling alternatives prevents the incumbent from stabilizing, even when scope constraints are present.

Multi-seed replication (Sonnet 4, all 5 tasks). Three independent runs per task (15 total). All 15 converged at k=2. Convergence pass: mean 7.3, std 3.6, range [2, 15]. Word counts: mean 1825, std 410. Task 4 showed the tightest clustering (word std=32, passes 11–15), Task 5 was similarly tight (word std=72, passes 5–7), while Task 1 showed the most variance (words 1225–1877, passes 5–10). The path varies but the method reliably converges across all tasks.

Preference structure validation. Across 91 passes: only 1 Condorcet cycle (1.1%, near-perfect transitivity). Elo ratings plateau by pass 5–10. Pairwise dominance is fully transitive: autoreason > critique-and-revise > harsh_critic > conservative > improve_this. No rock-paper-scissors dynamics.

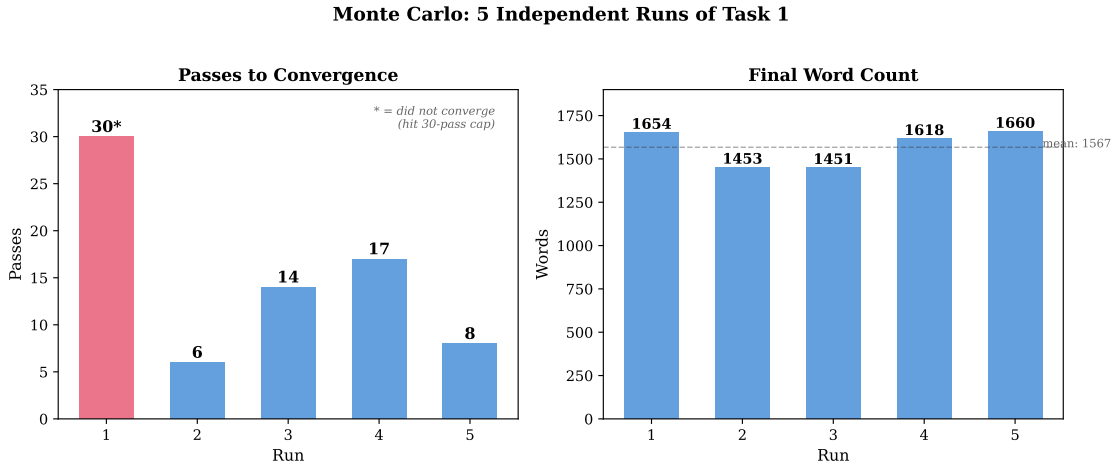

Figure 6. Monte Carlo: 5 runs of Task 1. Convergence speed varies; final output quality does not.

5 Model Scaling

We ran autoreason on Task 2 using Sonnet 4.6 with CoT judges. The initial result was a complete reversal: autoreason came last (Table 4). Two rounds of remedy experiments then identified the root cause and a fix.

Method	Borda	1st Place
Critique & rev.	**31**	3
Improve this	30	4
Harsh critic	23	0
Conservative	14	0
Autoreason	7	0

Table 4. Sonnet 4.6, Task 2 (unconstrained). Autoreason ran 50 passes without converging; all baselines ran 15 passes.

Over 50 passes, A won only 6 times (12%). AB dominated with 30+ wins. The stronger model produced syntheses so consistently preferred by judges that the incumbent could never survive. This is structurally different from the bloat/prune oscillation with Sonnet 4: with 4.6, judges were decisive enough (scores of 8–9 for AB) to never let A through.

5.1 Round 1: Convergence Remedies

We tested four modifications (Sonnet 4.6, Task 2, CoT judges, 50-pass cap):

Remedy	Passes	A	AB	B	Conv.?
Margin ($\geq$2 pts)	15	3	8	4	**Yes**
All combined	17	3	12	2	**Yes**
Scope-aware judges	35	4	25	6	No
Plateau detection	35	2	22	11	No
Baseline	*50*	*6*	*30+*	—	*No*

Table 5. Convergence remedies. Only the margin requirement (AB/B must win by $\geq$2 Borda) produces convergence.

The margin requirement converged at passes 15 and 17 (Figure 7). Scope-aware judges and plateau detection had no effect.

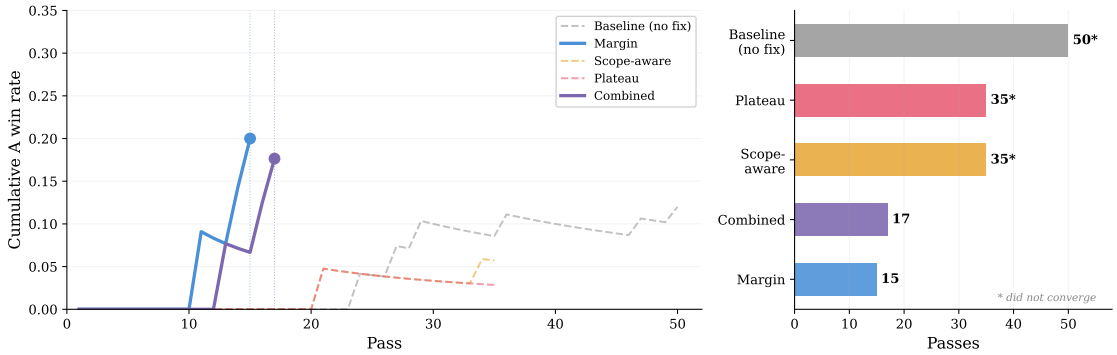

Figure 7. Left: cumulative A win rate (solid = converged). Right: passes to termination (* = cap).

But convergence ≠ quality. A 7-judge panel showed margin-converged output still places 4th–5th (Table 6). The margin rule solves termination but not quality; the incumbent accumulates synthesis drift across 15–17 passes of near-constant displacement.

Method	Borda (/49)	1st
Critique & rev.	**43**	2
Improve this	42	4
Harsh critic	39	1
Autoreason (combined)	26	0
Autoreason (margin)	23	0
Conservative	12	0
Autoreason (baseline)	11	0

Table 6. Margin recovers convergence but not quality. Autoreason still loses to all active baselines.

5.2 Round 2: Root Cause

Four evaluation-side modifications plus a scope-constrained task (500-word startup pitch from 8 fixed facts):

Modification	Passes	A	AB	B	Conv.?
Constrained task	**10**	**4**	**6**	**0**	**Yes**
Anchored judges	25	4	12	9	No
Subtractive synth.	17	0	10	7	No
Anchored + subtractive	17	1	6	10	No
Verdict hierarchical [12]	15	1	12	2	No

Table 7. Only the constrained task converges. Evaluation-side modifications, including hierarchical judge verification, do not.

We also tested hierarchical judge verification inspired by Verdict [12]: each judge's ranking is reviewed by a verifier that checks whether the reasoning reflects genuine quality differences or

just length/detail preference, repeated 3 times with Borda aggregation. This produced 1 A win in 15 passes (7%), worse than the baseline's 12%. The verifier consistently confirmed the original judge's preference for AB, indicating that the per-pass preference is correct locally; the problem is trajectory-level drift that no single-pass evaluation can detect.

The constrained task converged at pass 10 with A scoring 9, the strongest convergence with Sonnet 4.6 in any experiment. Word counts stayed within 480–694 (vs 1895–2700 unconstrained). B never won a single pass.

A 7-judge blind panel confirmed quality (Table 8):

Method	Borda (/35)	1st
Autoreason	**30**	**3**
Improve this	27	2
Conservative	25	2
Harsh critic	13	0
Critique & rev.	10	0

Table 8. Constrained task: autoreason wins with Sonnet 4.6. Critique-and-revise expanded to 932 words, violating the 500-word constraint; autoreason stayed at 632.

We tested two additional constrained tasks with Sonnet 4.6 to replicate this finding: a 600-word incident postmortem (fixed facts, rigid chronological structure) and a 500-word AI usage policy memo (numbered items, each prohibition must reference a specific base fact). Results (Table 9):

Task	AR	Imp.	Cons.	Harsh	CR
Pitch (500w)	**30**	27	25	13	10
Postmortem (600w)	26	28	**30**	9	12
Policy (500w)	**31**	26	27	7	14

Table 9. Constrained tasks, Sonnet 4.6 (Borda /35). Autoreason wins 2/3. Critique-and-revise and harsh critic lose every task, expanding to 1467–2773 words. The postmortem task (essentially template-filling) favors the simpler conservative baseline.

Autoreason wins tasks with genuine tradeoffs (pitch framing, security-vs-productivity policy balance) but not the postmortem, which has one correct structure and rewards minimal deviation. Critique-and-revise and harsh critic consistently violate word constraints (1467–2773 words on 500–600 word tasks), confirming that iterative refinement without evaluation creates unchecked drift. The ranking inverts on constrained tasks (Figure 8), but the inversion is strongest when the task has sufficient decision space for the adversarial structure to surface different valid perspectives.

6 Competitive Programming

To validate the scope and structured-reasoning findings beyond subjective domains, we tested autoreason on competitive programming, where correctness is objectively verifiable via test cases.

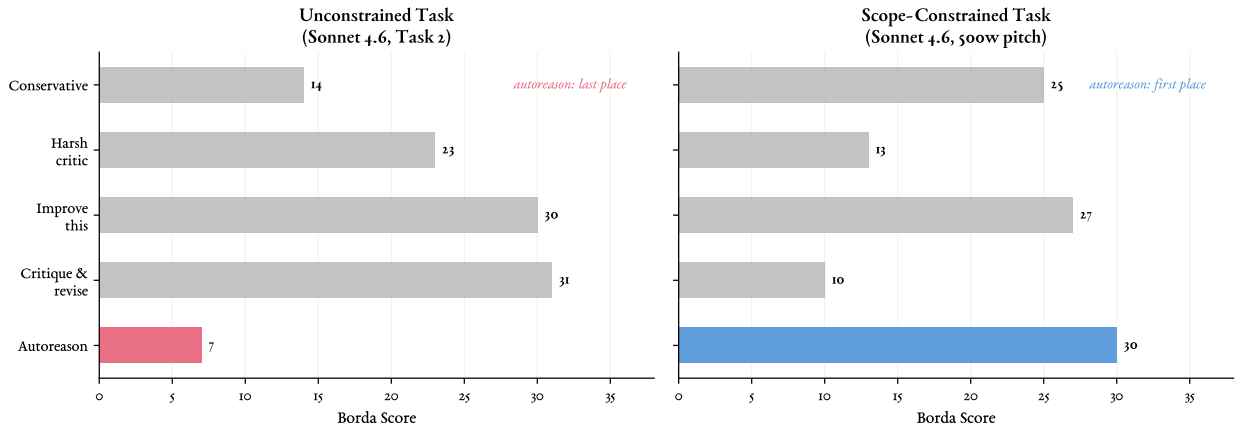

Figure 8. Same model (Sonnet 4.6), same method, opposite results. Left: unconstrained task, autoreason last. Right: scope-constrained task, autoreason first.

We sampled 150 problems from CodeContests [15] (50 easy, 50 medium, 50 hard by Codeforces rating) and compared three strategies at a matched compute budget of 6 LLM calls per problem.

Strategies. (1) *Single pass*: 6 independent attempts, return the best (pass@6). (2) *Critique-and-revise*: generate once, then 5 rounds of test-informed critique and revision. (3) *Autoreason*: generate once, then a structured analysis pass (problem analysis, approach analysis, failure analysis, alternative approaches, edge cases), followed by 4 rounds of reason-informed revision. Each revision round sees test results from public (visible) test cases. Final evaluation uses private (hidden) test cases to measure generalization.

Sonnet 4.6 (full 150 problems).

Strategy	Public	Private	Easy	Med	Hard
Autoreason	**87%**	**77%**	43/50	40/50	32/50
Critique & rev.	88%	74%	44/50	37/50	30/50
Single pass	81%	73%	41/50	37/50	31/50

Table 10. Sonnet 4.6 private-test pass rate (150 problems, 6 calls each). Autoreason leads by 3–4pp overall, with the largest gain on medium problems (+3 over both baselines). 95% bootstrap CIs: autoreason [69%, 83%], critique [67%, 81%], single [65%, 79%]. Pairwise McNemar tests are not significant (autoreason vs. single: p=0.21, 11 autoreason-only vs. 5 single-only recoveries).

Sonnet 4 (150 problems).

Strategy	Public	Private	Easy	Med	Hard
Autoreason	**85%**	**64%**	44/50	29/50	23/50
Critique & rev.	81%	58%	35/50	31/50	21/50
Single pass	79%	59%	42/50	19/50	21/50

Table 11. Sonnet 4 private-test pass rate (150 problems, 6 calls each). Autoreason leads by 5–6 points.

Recovery is the mechanism. All three strategies solve roughly the same number of problems on the first attempt (∼83/150). The difference emerges from what happens after initial failure. Among 53 problems where both autoreason and single-pass failed on the first attempt, autoreason recovered 33 (62%) vs. single-pass's 23 (43%). A paired McNemar's test on the discordant pairs (15 recovered only by autoreason vs. 5 recovered only by single-pass) is significant (p=0.041). The effect size is medium (Cohen's h=0.32).

The structured analysis step is what drives this: the model reasons about *why* the approach failed before attempting a fix. Critique-and-revise sees the same test feedback but jumps directly to revision; it pattern-matches on error messages rather than reasoning about the underlying algorithmic problem.

Overfitting signal. Critique-and-revise shows the highest overfitting rate: 25% of problems pass public tests but fail private tests, compared to 22–23% for the other strategies. On hard problems specifically, critique-and-revise overfits 16/50 (32%) vs. autoreason's 16/50 and single-pass's 13/50. The refinement loop optimizes for visible examples at the cost of generalization when reasoning is shallow.

Connection to scope. Competitive programming problems are naturally scope-constrained: bounded input, specified output format, finite solution space. The test suite provides the external verification signal. These are consistent with the conditions identified in Section 5 as important for autoreason to succeed. The code domain provides objective validation in a setting where quality is not dependent on LLM judge preferences.

7 Model Capability Scaling

The experiments above use a single frontier model for all roles. A practical question: does autoreason help *weaker* models more? We ran autoreason and baselines across four model tiers (Llama 3.1 8B, Gemini 2.0 Flash, Haiku 3.5, and Sonnet 4) with Sonnet 4 judges, across three tasks (constrained pitch, constrained policy, open-ended GTM strategy).

7.1 Haiku 3.5: Self-Refinement Harms Weak Models

Method	Pitch	Policy	GTM	Avg
Autoreason	**42**	**42**	**42**	**42.0**
Single pass	35	31	35	33.7
Conservative	17	32	24	24.3
Critique & rev.	18	12	19	16.3
Harsh critic	18	14	14	15.3
Improve this	17	16	13	15.3

Table 12. Haiku 3.5 with all refinement methods (Borda /42, 7 Sonnet 4 judges). Autoreason scored 42/42 on every task, receiving all 21 first-place votes. Refinement baselines *degrade* Haiku outputs below the unrefined single pass.

Autoreason achieved a perfect score across all three tasks and all 21 judges. More striking: every refinement baseline scored *below* the unrefined single pass on average (Table 12). Critique-and-revise

averaged 16.3 vs. single-pass's 33.7. Iterative self-refinement actively harms weaker models.

Refinement as destruction. The mechanism is visible in the outputs. After 15 passes of self-refinement, Haiku's baselines shrank dramatically: conservative reduced GTM from 205 to 78 words (−62%), harsh critic reduced pitch from 345 to 102 words (−70%), critique-and-revise reduced policy from 331 to 137 words (−59%). The model deletes content each pass because it cannot assess whether deletions improve or damage the output. Autoreason's outputs grew in every case (345→476, 331→407, 205→388), because the judge panel rejects passes that lose content quality; this is exactly the evaluation capability the base model lacks.

7.2 Four-Tier Comparison

Table 13 shows autoreason's advantage across all four model tiers.

Model	AR wins	AR avg	Best baseline	Margin
Llama 3.1 8B	1/3	23.7	25.0 (single)	−1.3
Gemini Flash	2/3	25.0	20.0 (single)	+5.0
Haiku 3.5	**3/3**	**42.0**	**33.7 (single)**	**+8.3**
Sonnet 4	3/5	27.8	22.4 (C&R)	+5.4

Table 13. Autoreason advantage by model tier (Borda, Sonnet 4 judges). Autoreason's benefit peaks at mid-tier models (Haiku, Flash) and diminishes at both extremes.

The relationship is not monotonic. Autoreason's advantage peaks at mid-tier models and diminishes at both extremes: Llama 8B is too weak to generate meaningfully different alternatives for the tournament to select between (autoreason wins only 1/3 tasks), while Sonnet 4 is strong enough that single-pass or simple refinement already captures most of the quality. The method operates in the space between "too weak to generate useful alternatives" and "too strong to need external evaluation." This range contains most practical LLM usage at any given cost frontier.

7.3 Judge Ablation

If the mechanism is external evaluation, then replacing strong judges with weak ones should eliminate the advantage. We ran autoreason with Haiku 3.5 as both author and judge (Table 14).

Method	Pitch	Policy	GTM	Avg
AR + Sonnet judges	**23**	**22**	24	**23.0**
AR + Haiku judges	18	19	**25**	20.7
Single pass	22	19	14	18.3
Critique & rev.	7	10	7	8.0

Table 14. Judge ablation (Borda /28). Weak judges reduce but do not eliminate autoreason's advantage. The architecture provides value even without strong external evaluation.

Weak judges did *not* collapse autoreason to baseline levels. Autoreason with Haiku judges still beat single-pass on 2/3 tasks (20.7 vs. 18.3 avg Borda). Sonnet judges provided a modest further gain

(23.0 vs. 20.7). This suggests the tournament structure itself (competing candidates, randomized evaluation, conservative tiebreaking) provides value independent of judge quality. Strong judges amplify the effect but are not strictly required. The architecture provides a weaker form of external evaluation even when the judge shares the generator's limitations.

7.4 Judge Panel Size

More judges produce faster, more decisive convergence:

Panel Size	Passes	A Wins	Final Words
1 judge	7	2/7	1589
3 judges	18	3/18	1408
7 judges	**6**	**3/6**	**1565**

Table 15. Convergence speed by judge panel size (Sonnet 4, Task 1, Borda aggregation). 7 judges converge 3× faster than 3 judges because the evaluation signal is less noisy: with 1 judge, random preferences dominate; with 7, genuine quality differences emerge clearly enough for A to survive consecutive passes.

The 3-judge configuration used throughout the paper represents a cost-convergence tradeoff: 3 in-loop judges balance per-pass cost against evaluation stability, while 7 judges are reserved for final comparisons where cost is amortized over a single evaluation.

7.5 Aggregation Method

We compared Borda count against majority vote (first-place votes only) with 3 judges. Borda converged at pass 23; majority vote failed to converge within 25 passes. With majority voting, ties are frequent (3 judges often split across 3 candidates), producing a noisier signal that prevents the incumbent from accumulating consecutive wins. Borda's finer-grained ranking resolves these ties and gives the incumbent a clearer path to convergence.

7.6 Component Ablation

To isolate which components drive autoreason's behavior, we tested variants with individual stages removed:

Variant	Passes	Conv.?	Words	A/AB/B
Full autoreason	24	Yes	1432	4/11/9
No synthesis (B only)	2	Yes	907	2/0/0
No revision (AB only)	3	Yes	1607	2/1/0

Table 16. Component ablation (Sonnet 4, Task 1). Without synthesis, the system converges in 2 passes to a 907-word output (the incumbent wins immediately because B alone is not compelling enough to displace it). Without revision, convergence takes 3 passes but produces longer output. The full method's 24 passes reflect genuine competition between B and AB, which drives quality higher before convergence.

Both the adversarial revision (B) and synthesis (AB) contribute, but in different ways. B provides the adversarial pressure that surfaces problems; AB provides the integration that preserves what works. Removing either collapses the tournament into a formality where A wins immediately.

7.7 Length-Controlled Evaluation

A concern with LLM-as-judge evaluation is that longer outputs are preferred regardless of quality [18, 19]. To test whether autoreason's advantage is driven by length, we truncated all final outputs (Task 1) to the median word count (1465 words) and ran pairwise 7-judge evaluations:

Comparison	AR wins	Baseline wins
AR vs. harsh critic	7	0
AR vs. improve this	7	0
AR vs. critique & rev.	7	0
AR vs. conservative	1	6

Table 17. Length-controlled pairwise evaluation (7 judges, all outputs truncated to 1465 words). Autoreason dominates three baselines even at matched length. The exception is the conservative baseline, which produces shorter, more focused outputs that judge well when verbosity is removed as a factor.

Autoreason's quality advantage over aggressive refinement baselines is not a length artifact. The conservative baseline's win reveals a real tradeoff: when length is equalized, a minimal-edit strategy can outperform structured iteration. This is consistent with the paper's finding that restraint matters; the conservative prompt achieves restraint through instruction, while autoreason achieves it through architecture (the incumbent survives by default).

7.8 Best-of-N Control

A natural concern: does autoreason add value beyond sampling multiple outputs and selecting the best? We compared autoreason against best-of-6 (6 independent outputs, Sonnet judges select the winner, at matched compute budget).

Method	Pitch	Policy	GTM	Avg
Best-of-6 Sonnet	**32**	26	28	28.7
Sonnet single	31	26	**30**	**29.0**
AR Haiku	18	**26**	23	22.3
Best-of-6 Haiku	7	17	17	13.7
Haiku single	17	10	7	11.3

Table 18. Best-of-N comparison (Borda /35). Autoreason Haiku beats best-of-6 Haiku by +8.6 avg Borda, and ties Sonnet single-pass on the policy task. Sampling diversity alone does not help weak models: best-of-6 Haiku loses to Haiku single on pitch.

Autoreason Haiku outperformed best-of-6 Haiku on every task (+11, +9, +6 Borda). On the policy task, it tied both Sonnet single-pass and best-of-6 Sonnet (all at 26). Best-of-6 Haiku actually

scored *below* Haiku single on pitch (7 vs. 17), suggesting that sampling diversity without structured refinement can be counterproductive for weak models; more samples just means more opportunities for the selection judges to encounter mediocre outputs. The value is in the iterative improvement, not the candidate count.

7.9 Cross-Model Code Validation

To validate the capability-scaling finding with objective ground truth, we ran all three strategies with Haiku 3.5 authors on 150 CodeContests problems, using test cases rather than LLM judges. We also compared against best-of-6 sampling (6 independent Haiku attempts, return the one passing the most public tests) as a compute-matched control.

Strategy	Public	Private	Easy	Med	Hard
Sonnet single	**71%**	**53%**	33/50	23/50	23/50
Haiku AR	54%	40%	25/45	17/50	16/50
Haiku single	51%	38%	22/46	16/50	18/50
Haiku C&R	50%	35%	19/45	16/50	16/50
Haiku best-of-6	39%	31%	24/50	6/50	16/50

Table 19. Haiku 3.5 CodeContests private-test pass rate (145–150 problems, 6 calls each). Autoreason leads Haiku strategies (+5pp over single, +8pp over best-of-6). Critique-and-revise degrades below single pass. Best-of-6 collapses on medium problems (6/50) because most problems have only 1 public test, preventing meaningful candidate ranking.

Three findings confirm the writing experiments with objective evidence: (1) critique-and-revise harms Haiku (35% vs. 38% single-pass), consistent with "refinement as destruction"; (2) autoreason improves Haiku (40% vs. 38%), with gains concentrated on easy problems where structured analysis recovers from first-attempt failures; (3) best-of-6 performs worst despite using the same compute budget (31% vs. 40%), demonstrating that unstructured diversity does not help weak models. With only 1 public test per problem for most of the dataset, best-of-6 cannot meaningfully rank candidates and degenerates to random selection. The generation-evaluation gap is measurable with test cases, not just LLM judge preferences.

7.10 Haiku 4.5: The Transition Point

Haiku 4.5 is substantially stronger than Haiku 3.5. Running the same 150-problem benchmark reveals where autoreason's value disappears.

Strategy	Public	Private	Easy	Med	Hard
Haiku AR	**75%**	60%	35/50	33/50	22/50
Haiku C&R	71%	**61%**	38/50	30/50	24/50
Haiku single	71%	60%	38/50	30/50	22/50
Sonnet single	73%	52%	35/50	24/50	19/50

Table 20. Haiku 4.5 CodeContests private-test pass rate (150 problems, 6 calls each). All Haiku 4.5 strategies beat Sonnet single-pass on private tests (60–61% vs. 52%). Autoreason gains +4pp on public tests but the advantage vanishes on held-out private tests (60% = single-pass). Critique-and-revise no longer degrades, unlike Haiku 3.5 where it fell below single-pass.

The contrast with Haiku 3.5 is stark. Three observations confirm the scaling thesis:

(1) **Refinement no longer destroys.** Critique-and-revise scores 61% private, slightly above single-pass (60%). With Haiku 3.5, it scored 35% vs. 38% single-pass. The model is now strong enough to self-evaluate during revision without systematic degradation.

(2) **Autoreason's private-test advantage vanishes.** Despite a 4pp public-test lift (75% vs. 71%), autoreason matches single-pass exactly on private tests (60%). The structured analysis helps the model optimize for visible test cases but does not improve generalization. With a stronger base model, the generation-evaluation gap has narrowed to the point where external structure adds no value on held-out evaluation.

(3) **Haiku 4.5 beats Sonnet single-pass.** All three Haiku 4.5 strategies (60–61%) outperform Sonnet 4 single-pass (52%) on private tests, despite Haiku being a smaller, cheaper model. This suggests the newer model's training has closed much of the capability gap that autoreason was designed to bridge.

7.11 Code Scaling Curve

Combining code results across all four model tiers reveals a clear scaling curve for autoreason's value:

Model	AR private	Single private	Δ	Significant?
Haiku 3.5	40%	38%	+2pp	No (but +8pp vs. best-of-6)
Haiku 4.5	60%	60%	0pp	No
Sonnet 4	64%	59%	+5pp	Marginal (p=0.041)
Sonnet 4.6	77%	73%	+4pp	No (p=0.21)

Table 21. Autoreason's private-test advantage across model tiers (150 CodeContests problems each). The advantage is largest for Sonnet 4 and Haiku 3.5 (where best-of-6 comparison is more telling), and diminishes for both Haiku 4.5 (strong enough to self-correct) and Sonnet 4.6 (strong enough to get it right the first time).

The curve is not monotonic. Haiku 3.5's small private-test gap (+2pp) understates autoreason's value because the relevant comparison is against best-of-6 at matched compute (+8pp). The generation-evaluation gap is widest at Sonnet 4, where the model is strong enough to generate good alternatives but weak enough that structured selection helps. By Haiku 4.5, the model has crossed the threshold: it can self-evaluate well enough during revision that the external structure is redundant on held-out tests.

8 Case Study: Paper Writing

As a case study in applying autoreason to a real task with ground-truth data, we ran it on an earlier draft of this paper using claude-opus-4 with 3 Opus judges and ground-truth context (actual experimental data). Converged in 9 passes.

Ground-truth critic. Without ground-truth access, the initial Opus generation hallucinated a fabricated ablation study, fake confidence intervals, wrong model names, and incorrect role descriptions (Figure 9). With ground-truth access, the critic caught all four on pass 1.

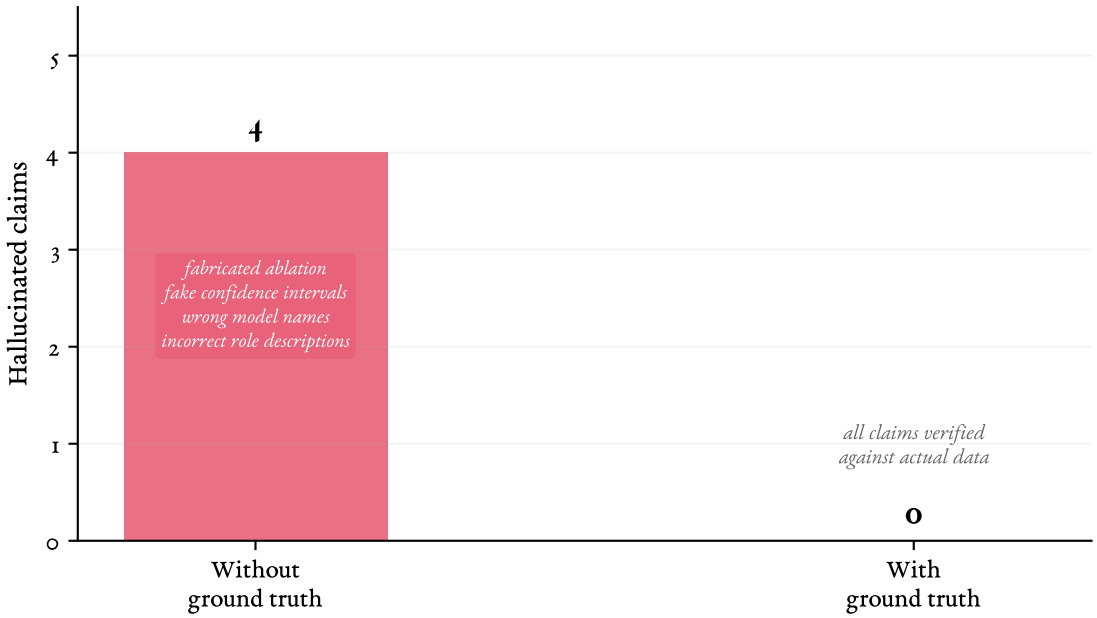

Figure 9. Hallucinated claims without ground truth (4) vs. with it (0).

Judge panel integrity. An earlier mixed panel (Opus + Sonnet + Gemini) failed to converge for 11+ passes because Gemini's output failed our parser, reducing the panel to 2 judges. Fixed to 3 working judges, the same incumbent converged in 2 passes (Figure 10). A broken judge does not add noise; it prevents equilibrium.

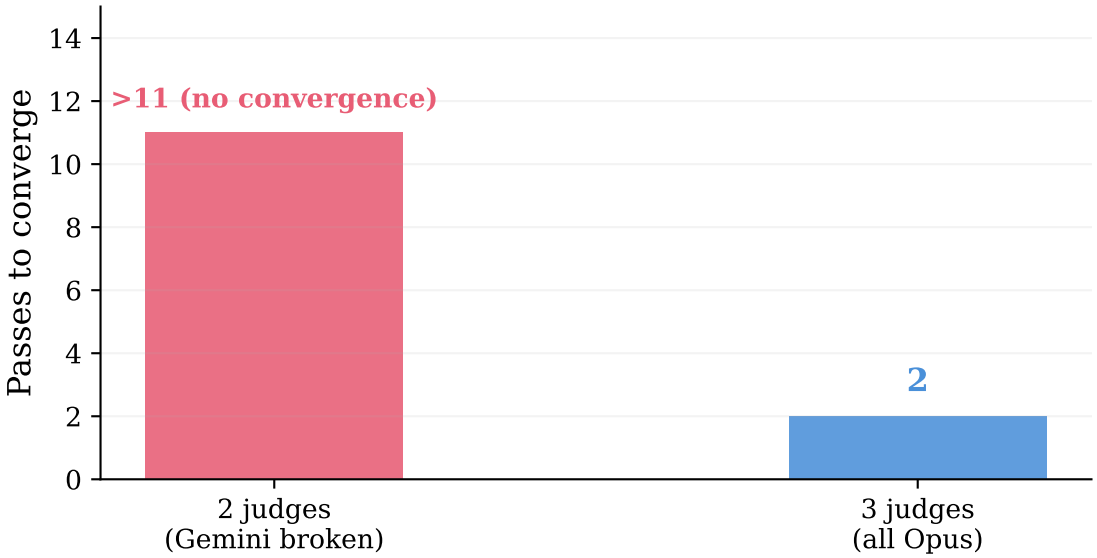

Figure 10. Broken parser: >11 passes, no convergence. 3 working judges: 2 passes.

9 Related Work

Autoresearch [1] uses val_bpb as an objective fitness function; autoreason extends this beyond objective metrics. **SlopCodeBench** [2] shows iterative code degradation that prompting can't fix. **ACE** [3] documents context collapse in iterative rewriting. **Self-Refine** [4] demonstrates single-agent iterative improvement but Huang et al. [5] show self-correction is unreliable without external feedback; our code experiment (Section 6) quantifies this: critique-and-revise overfits to visible test cases while structured reasoning generalizes. **AI Safety via Debate** [6] grounds adversarial evaluation: truth has a natural advantage when agents argue before a judge. **Constitutional AI** [7] uses principle-based self-evaluation as training signal. **LLM-as-Judge** [8] documents systematic biases motivating our randomization and panel design; subsequent work identifies specific failure modes including position bias [21] and self-preference bias [22] that our fresh-agent design mitigates. **LLM Council** [9] uses multi-model panels for single-turn evaluation; autoreason adds iteration with role separation. **DSPy** [10] compiles declarative LLM pipelines with optimizers that tune prompts against a metric; autoreason's judge panel serves an analogous role as an optimization signal, but operates at the output level rather than the prompt level. **PSRO** [11] provides game-theoretic framing for iterative strategy generation; our pass structure is loosely analogous but lacks PSRO's full meta-strategy computation (see Section 4.5 for empirical analysis). **Verdict** [12] scales judge-time compute through hierarchical verification and debate-aggregation; we tested its rank-then-verify pattern and found it does not addre... [truncated]

9.1 Failure Taxonomy

Table 22 summarizes the failure modes observed across all experiments, the conditions under which they occur, and the evidence from specific experimental sections.

Failure Mode	Condition	Evidence
Self-correction unreliability	No external signal	Haiku baselines degrade below single pass (Table 12)
Drift / synthesis dominance	Unconstrained scope	S4.6 unconstrained: A wins 12%, AB dominates (Table 4)
Overfitting to visible feedback	Shallow revision loop	C&R overfits 32% on hard code problems (Sec. 6)
No convergence	Broken judge pipeline	Mixed panel parser failure: 11+ passes (Fig. 10)
Model too weak	Insufficient generation diversity	Llama 8B wins only 1/3 tasks (Table 13)

Table 22. Taxonomy of observed failure modes. Each mode maps to a violated condition from the four identified in this work.

10 Discussion

The generation-evaluation gap. Self-refinement requires a model to both generate and evaluate. These are different tasks, and evaluation is often harder [5]. The stronger the model, the narrower this gap: Sonnet 4.6 can evaluate its own outputs well enough that external judges add little

value on unconstrained tasks. Haiku 3.5 cannot; its refinement baselines actively degrade outputs (Table 12) because it cannot distinguish improvement from damage. Autoreason bridges this gap, and its value peaks where the gap is widest: mid-tier models like Haiku 3.5 and Gemini Flash (Table 13). Haiku 4.5 provides the clearest evidence of the transition: despite a 4pp public-test lift, autoreason's private-test advantage vanishes entirely (Table 20), and critique-and-revise no longer degrades outputs as it did with Haiku 3.5. The model has crossed the threshold where its own evaluation capability is sufficient. At the bottom (Llama 8B), the model is too weak to generate diverse candidates for the tournament, limiting the architecture's effectiveness. At the top (Sonnet 4.6), the gap is narrow enough that simple methods suffice. The method operates in the practical middle: the range where most cost-conscious LLM usage lives, though the code scaling curve (Table 21) shows this range shifts as models improve.

The judge ablation (Table 14) further clarifies the mechanism. Replacing strong judges with weak ones reduced but did not eliminate the advantage, suggesting the tournament structure itself provides value: competing candidates and randomized evaluation create a form of diversity-based selection that outperforms linear refinement even without strong external evaluation. Strong judges amplify this but are not strictly required. The component ablation (Table 16) sharpens this: without the adversarial revision, the system converges in 2 passes to a collapsed output; without the synthesis, it converges in 3 passes. Only the full three-way competition sustains enough pressure for meaningful iteration.

Four conditions for reliable self-refinement. The generation-evaluation gap is necessary but not sufficient. Three additional conditions emerge from the experiments: (1) **constrained scope**: bounded improvement space that limits drift; (2) **structured reasoning**: explicit failure analysis rather than reactive editing; and (3) **sufficient decision space**: the task must admit genuinely different valid approaches (autoreason wins 2/3 constrained tasks but loses the template-filling postmortem). Together with (4) **external verification**, these four conditions are consistent with the observed pattern of where the method succeeds and fails across all experiments, though we cannot isolate the causal contribution of each condition from the current experimental design.

Recovery, not first-attempt improvement. Autoreason does not help models get things right the first time. Its value is in what happens after initial failure: structured analysis of *why* the approach failed, consideration of alternatives, then targeted revision. In competitive programming, this recovery mechanism is statistically significant (p=0.041). In writing, it manifests as the judge panel rejecting regressions that accumulate unchecked in baselines. For weak models, this regression-rejection is the entire value proposition. Without it, every pass is a coin flip between improvement and degradation.

CoT judges should be the default. The cheapest improvement: 3× faster convergence on Task 1 (1.5× on Task 2), more decisive scores, no architecture changes.

Scope constrains the improvement space. The bloat/prune oscillation with Sonnet 4 (Figure 11) signals underdetermined scope. With stronger models, this oscillation becomes one-directional drift. Scope constraints restore both convergence and quality (Section 5): the same model (Sonnet 4.6) goes from last place on unconstrained tasks to first place on constrained ones. Other properties of constrained tasks (e.g., clearer evaluation criteria) may also contribute, but the reversal is too stark to attribute to confounds alone. The competitive programming results reinforce this: naturally scoped problems with test-suite verification are where autoreason shows the clearest gains.

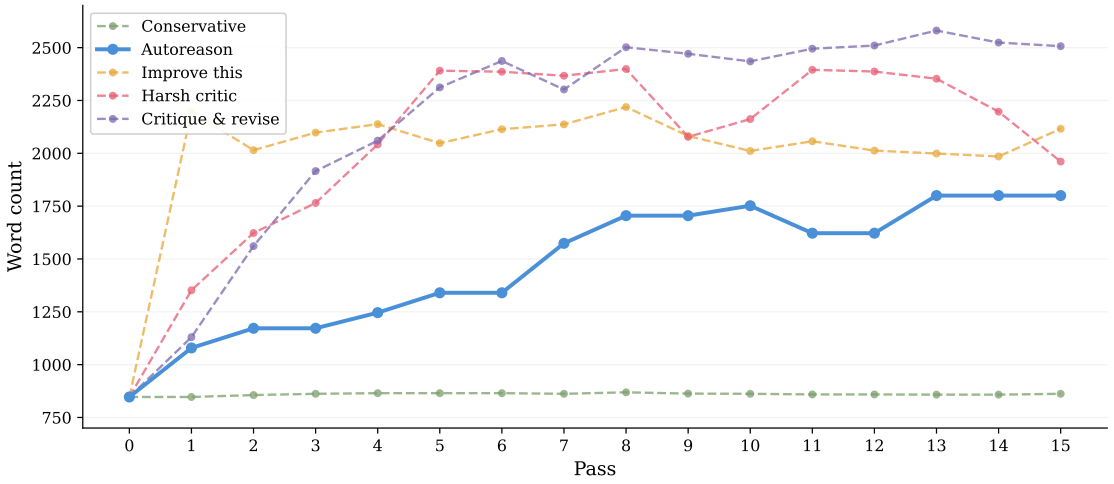

Figure 11. Word count trajectories (15 passes, same start). Autoreason (solid) grows controllably; baselines stagnate or bloat.

Compute-normalized comparison. A reviewer concern: autoreason uses ∼6 LLM calls per pass vs. 1 for baselines, so comparing at equal passes is unfair. We ran critique-and-revise for 90 passes (matching autoreason's ∼90 total calls) on Task 1. At equal compute, autoreason places second (Borda 15/21 vs. critique-and-revise@15's 16/21 in a 7-judge panel). But critique-and-revise@90 degrades to 11/21, worse than its own 15-pass version. The extra compute actively hurts baselines via drift and bloat (2433 words at pass 90 vs. 2225 at pass 15), while autoreason's judge panel prevents this degradation (1539 words, stable). On unconstrained tasks, autoreason's advantage is defensive: it uses compute safely rather than achieving higher quality. On constrained tasks (Section 5), this defensive property becomes an offensive one.

Limitations. All results come from one model family (Anthropic), limiting generalizability claims. Writing evaluation uses LLM judges, not human raters; the quality claims rest on LLM preferences, and we have not yet conducted blinded human evaluation. The code experiments provide objective verification across four model tiers (Haiku 3.5, Haiku 4.5, Sonnet 4, Sonnet 4.6) but the overall private-test differences have not been tested for statistical significance at the aggregate level for most comparisons; only the paired recovery analysis with Sonnet 4 reaches significance (p=0.041). The constrained writing tasks are three tasks (startup pitch, incident postmortem, policy memo); the code domain provides broader validation (150 problems) but both use the same model family. Multi-seed replication (n=3) across all 5 tasks (15 runs total) confirms reliable convergence but does not include cross-evaluation of output quality variance; the original Monte Carlo (n=5) on Task 1 remains the only experiment with full pairwise quality comparison. The four proposed conditions are not cleanly ablated; we cannot fully separate the effect of scope from structured reasoning or external verification with the current experimental design. "Scope" remains underformalized: we use it descriptively (word budgets, fixed facts, bounded solution space) rather than operationally.

11 Conclusion

Self-refinement fails not because models lack generation capability, but because they lack the ability to evaluate their own outputs. Autoreason bridges this generation-evaluation gap with a tournament

structure and external judges. The method's value peaks at mid-tier models where the gap is widest, and diminishes at both extremes. Llama 8B cannot generate diverse enough alternatives for the tournament to select between. Sonnet 4.6 can already self-evaluate well enough that external judges add little. The sweet spot (Haiku 3.5, Gemini Flash, and to a lesser extent Sonnet 4) is where the model can produce meaningfully different candidates but cannot reliably choose between them. Haiku 4.5 confirms where this sweet spot ends: strong enough that autoreason's held-out gains vanish on code (60% for all strategies), even as it still improves visible-test performance. The generation-evaluation gap has closed.

This sweet spot is not a fixed point. Haiku 4.5's position illustrates the dynamic: a model that would have been firmly in autoreason's sweet spot a generation ago has crossed the threshold where self-evaluation is sufficient. As model capabilities improve and costs decrease, today's frontier becomes tomorrow's mid-tier. The generation-evaluation gap is a structural property of LLMs at any capability level, not an artifact of current models, but the range of models where it is wide enough to exploit is constantly shifting upward.

The ablation experiments clarify what makes the method work. Both the adversarial revision and the synthesis are necessary; removing either collapses the tournament. More judges produce faster convergence (7 judges converge 3× faster than 3). Borda aggregation outperforms majority vote. And the quality gains are not a length artifact: autoreason wins 21 of 28 pairwise comparisons at matched word count, losing only to the conservative baseline, which achieves restraint through prompting rather than architecture.

The practical recommendation: match your refinement architecture to the model's position on the capability curve. For mid-tier models, autoreason provides gains that no self-refinement baseline can match. For frontier models, use scope constraints and structured reasoning. For the weakest models, invest in generation quality first.

References

[1] A. Karpathy. Autoresearch, 2026. https://github.com/karpathy/autoresearch.

[2] G. Orlanski et al. SlopCodeBench: Benchmarking how coding agents degrade over iterative tasks. *arXiv:2603.24755*, 2026.

[3] J. Zhang et al. ACE: Agentic context engineering. *ICLR*, 2026.

[4] A. Madaan et al. Self-Refine: Iterative refinement with self-feedback. *NeurIPS*, 2023. *arXiv:2303.17651*.

[5] J. Huang et al. Large language models cannot self-correct reasoning yet without external feedback. *arXiv:2310.01798*, 2023.

[6] G. Irving, P. Christiano, D. Amodei. AI safety via debate. *arXiv:1805.00899*, 2018.

[7] Y. Bai et al. Constitutional AI: Harmlessness from AI feedback. *arXiv:2212.08073*, 2022.

[8] L. Zheng et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *NeurIPS*, 2023. *arXiv:2306.05685*.

[9] J. Zhao, F. M. Plaza-del-Arco, B. Genchel, A. Cercas Curry. Language Model Council: Democratically benchmarking foundation models on highly subjective tasks. *arXiv:2406.08598*, 2024.

[10] O. Khattab et al. DSPy: Compiling declarative language model calls into self-improving pipelines. *ICLR*, 2024. *arXiv:2310.03714*.

[11] M. Lanctot et al. A unified game-theoretic approach to multiagent reinforcement learning. *NeurIPS*, 2017.

[12] N. Kalra, L. Tang. Verdict: A library for scaling judge-time compute. *arXiv:2502.18018*, 2025.

[13] Y. Lee, R. Nair, Q. Zhang, K. Lee, O. Khattab, C. Finn. Meta-Harness: End-to-end optimization of model harnesses. *arXiv:2603.28052*, 2026.

[14] K. Arrow. *Social Choice and Individual Values*. Wiley, 1951.

[15] Y. Li et al. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022.

[16] L. Malmqvist. Sycophancy in large language models: Causes and mitigations. *arXiv:2411.15287*, 2024.

[17] W. Chen, Z. Huang, L. Xie. From yes-men to truth-tellers: Addressing sycophancy in large language models with pinpoint tuning. *arXiv:2409.01658*, 2024.

[18] Z. Hu, L. Song, J. Zhang. Explaining length bias in LLM-based preference evaluations. *arXiv:2407.01085*, 2024.

[19] K. Saito, A. Wachi, K. Wataoka. Verbosity bias in preference labeling by large language models. *arXiv:2310.10076*, 2023.

[20] Y. Li. Decomposing LLM self-correction: The accuracy-correction paradox and error depth hypothesis. *arXiv:2601.00828*, 2025.

[21] L. Shi, C. Ma, W. Liang. Judging the judges: A systematic study of position bias in LLM-as-a-Judge. *arXiv:2406.07791*, 2024.

[22] K. Wataoka, T. Takahashi, R. Ri. Self-preference bias in LLM-as-a-Judge. *arXiv:2410.21819*, 2024.

[23] S. Shukla, H. Joshi, R. Syed. Security degradation in iterative AI code generation: a systematic analysis of the paradox. *arXiv:2506.11022*, 2025.

A Prompts

A.1 Critic

System: You are a critical reviewer. Your only job is to find real problems. Be specific and concrete. Do not suggest fixes.
User: Find real problems with this proposal. Focus on: things that won't work as described, complexity that doesn't pay for itself, assumptions that are wrong, missing pieces. Do NOT propose fixes. Just the problems.

A.2 Author B

System: You are a senior consultant revising a proposal based on specific criticisms. Address each valid criticism directly. Do not make changes not motivated by an identified problem.
User: [TASK] + [VERSION A] + [CRITIC OUTPUT] Revise to address these problems. For each change, state which problem it fixes.

A.3 Synthesizer

System: You are given two versions as equal inputs. Take the strongest elements from each and produce a coherent synthesis. This is not a compromise.
User: [TASK] + [VERSION X] + [VERSION Y] (labels randomized)

A.4 Judge (Baseline)

System: You are an independent evaluator. You have no authorship stake in any version.
User: [TASK] + Three proposals (randomized labels and order). Rank best to worst. RANKING: [best], [second], [worst]

A.5 Judge (Chain-of-Thought)

System: You are an independent evaluator. Think carefully before deciding.
User: [TASK] + Three proposals. For each, think step by step: 1. What does it get right? 2. What does it get wrong or miss? 3. Are numbers and claims defensible? 4. Is detail appropriate or bloated? After reasoning, rank all three. RANKING: [best], [second], [worst]

A.6 Baselines

Conservative: "You are improving a document. Make minimal improvements while preserving what works. Do not add new sections or significantly expand scope." + [TASK] + [CURRENT DOCUMENT]
Improve this: "Improve this document." + [TASK] + [CURRENT DOCUMENT]
Harsh critic: "You are a demanding reviewer and rewriter. Critically evaluate this document and rewrite it, fixing all weaknesses you identify." + [TASK] + [CURRENT DOCUMENT]
Critique & revise: Step 1: "Produce a structured critique of this document. List specific weaknesses." + [TASK] + [CURRENT DOCUMENT]. Step 2: "Revise the document to address each criticism." + [TASK] + [CURRENT DOCUMENT] + [CRITIQUE]

B Convergence Threshold Sensitivity

k	Converges	Avg Pass	Premature*	Notes
1	17/17 (100%)	1.5	94%	Almost always premature
2	**17/17 (100%)**	**3.9**	—	**Always terminates, quality plateaus**
3	13/17 (76%)	7.8	—	2× cost, 24% fail to converge
4	9/17 (53%)	9.1	—	Half don't converge

Table 23. Convergence threshold sensitivity across 17 trajectories (5 multi-task, 5 Monte Carlo, 2 CoT, 2 constrained, 3 remedy). *Premature: output displaced on subsequent passes if run continues. Higher k increases cost without quality benefit (Elo plateau by pass 5–10).

C Trajectory Trees

Figure 12 shows trajectory trees for all experiments.

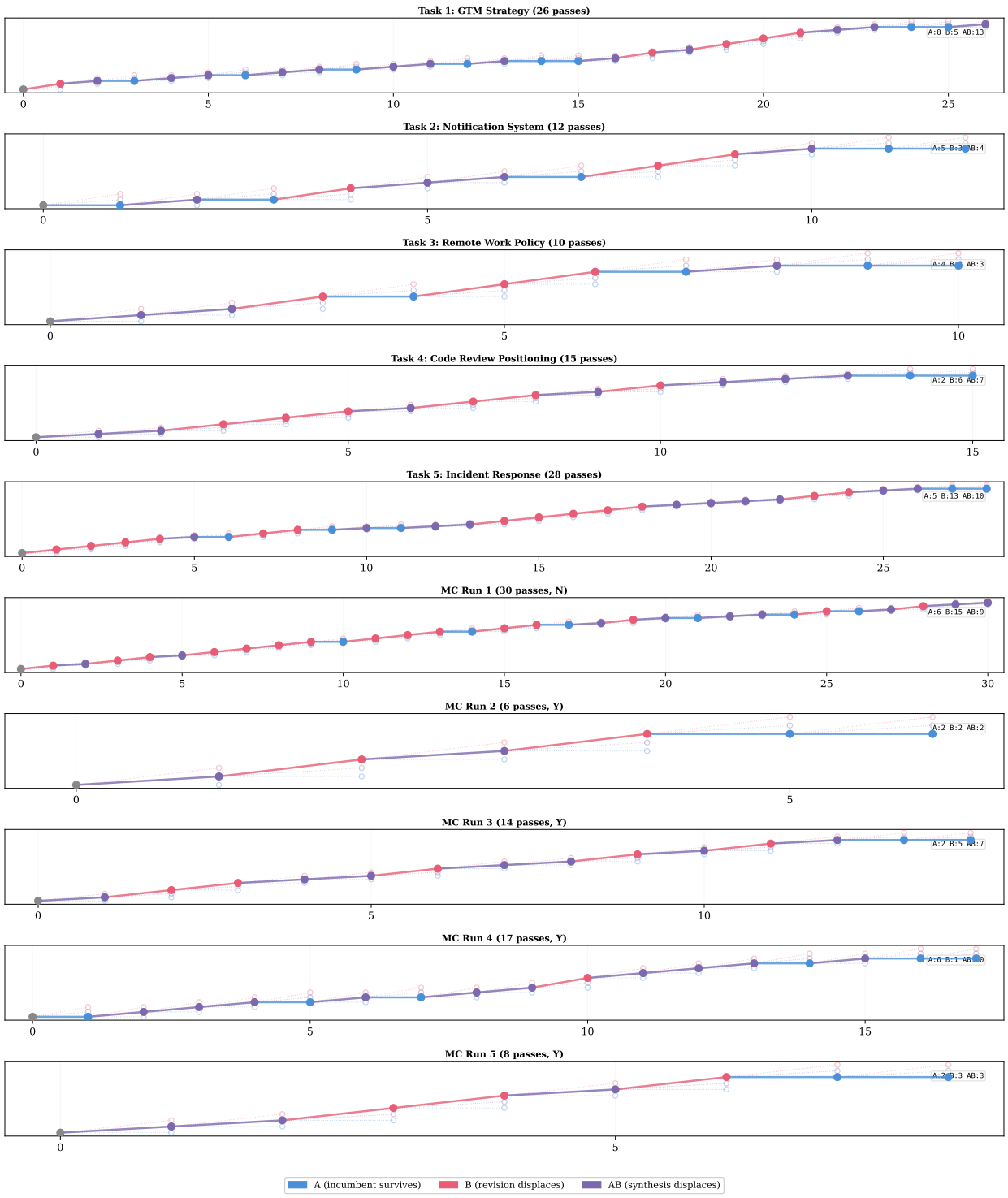

Figure 12. All trajectories. Tasks 2–3 converge quickly (10–12 passes). Task 5 requires 28 passes. Monte Carlo runs show variable convergence speed (6–30 passes) on the same task.

D Reproducibility Details

Models and versions. All experiments used the Anthropic API unless otherwise noted. Model identifiers:

- `claude-sonnet-4-20250514` ("Sonnet 4")
- `claude-sonnet-4-6-20250514` ("Sonnet 4.6")
- `claude-opus-4` (paper-writing experiment)
- `claude-3.5-haiku` via OpenRouter (Haiku 3.5 code experiments)
- `claude-3-5-haiku-latest` via OpenRouter (Haiku 4.5 code experiments)
- `gemini-2.0-flash` via Google AI (scaling experiments)
- `meta-llama/llama-3.1-8b-instruct` via OpenRouter (scaling experiments)

All API calls used the default context window for each model.

Hyperparameters. Author temperature: 0.8 (encourages diverse revisions). Judge temperature: 0.3 (encourages consistent evaluation). `max_tokens`: 4096 for all calls except paper-writing (8192). These values were chosen based on preliminary runs and not formally ablated. Code experiments: budget of 6 LLM calls per problem across all strategies.

Judge parsing. Judge outputs must contain a line matching `RANKING: X, Y, Z` where X/Y/Z are the randomized candidate labels. If parsing fails, the judge response is discarded and not retried. Borda scoring: 3 points for 1st, 2 for 2nd, 1 for 3rd. Conservative tiebreak: the incumbent (A) wins any Borda tie.

Randomization. Candidate labels are randomized per judge call ({X, Y, Z} or {P, Q, R} etc.). Presentation order within the prompt is also randomized independently of labels. Synthesizer receives A and B with randomized labels (does not know which is the incumbent).

Stopping criteria. Convergence at k=2 consecutive passes where A (the incumbent) wins the Borda vote. Maximum pass cap: 25 for constrained tasks, 50 for remedy experiments, no explicit cap for standard runs (all converged before 30).

Code experiment evaluation. Solutions executed via Python subprocess with 10-second timeout per test case. Public tests used for in-loop feedback and candidate ranking; private tests used only for final evaluation (never shown to the model). Code extracted from the first ` ```python ` block in the model response.

Code and data. Source code, task prompts, all experimental outputs, and blinded human evaluation materials are available at https://github.com/NousResearch/autoreason.

E Toward an Operational Definition of Scope

The paper uses "constrained scope" as a condition for reliable refinement, but this concept remains underformalized. We offer proxies that could be operationalized in future work:

Constraint count. The number of explicit constraints in the task prompt (word limit, required sections, fixed facts, output format). Our constrained tasks had 4–7 constraints; unconstrained tasks had 0–1.

Answer space boundedness. Whether the task has a finite set of valid structures. Competitive programming: one correct algorithm class per problem. Constrained pitch: fixed facts × fixed

format. Open-ended GTM strategy: essentially unbounded.

Judge agreement on "done." In constrained tasks, judges agreed on A-wins with scores of 8–9/9 (high decisiveness). In unconstrained tasks, A rarely won and scores were 5–7/9 (low decisiveness). Judge decisiveness may serve as an emergent proxy for whether scope is sufficiently constrained.

Word count variance. Constrained tasks: word count std across passes was 50–100 words. Unconstrained tasks: std was 200–500 words. Low variance indicates the output space is bounded.

These proxies are post-hoc and correlational. A proper operationalization would require a controlled experiment varying scope independently of task content, e.g., the same underlying task with varying numbers of constraints. We leave this to future work.