CoT^2-META: BUDGETED METACOGNITIVE CONTROL FOR TEST-TIME REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Effective test-time reasoning requires deciding which partial trajectories to expand, which errors to repair, and when to conclude a search. We introduce CoT^2-Meta, a training-free framework that combines strategy-conditioned thought generation, tree-structured search, online process evaluation, and state-dependent control over expansion, pruning, repair, stopping, and abstention. A compact meta-state links intermediate reasoning quality to the allocation of a total-token budget. At $8B_{\text{greedy}}$, CoT^2-Meta achieves 92.8% on MATH-500, 90.4% on GPQA-Diamond, 75.8% on BBEH, and 56.7% Pass@1 on LiveCodeBench with Claude-4.5-Opus, and leads the evaluated baselines on all four tasks with Llama-3.1-8B-Instruct. Holding the inference components fixed and changing only action selection raises four-task macro accuracy by 2.0 points over a threshold-only policy on each backbone. A 900-state counterfactual audit associates this gain with higher best-action agreement and lower decision regret. Evaluator interventions, candidate-coverage audits, and token-matched repair comparisons connect the end-to-end gains to intermediate signal quality and targeted trajectory correction. Nested confidence-prefix evaluation further characterizes selective prediction separately from additional-budget fallback. These results establish an empirical case for explicit metacognitive control as a design principle for budgeted test-time reasoning.

1 INTRODUCTION

Additional inference-time computation creates opportunities for language models to solve difficult reasoning problems. Chain-of-thought prompting exposes intermediate steps, self-consistency aggregates multiple trajectories, and structured search explores alternative continuations (Wei et al., 2022; Wang et al., 2022; Yao et al., 2023). The value of these computations depends on the state in which they are performed. Extending a sound partial derivation, revising a localized error, and abandoning an unproductive branch serve different purposes, even when each consumes the same token allocation.

Process supervision and verifiers supply evidence about intermediate and terminal reasoning quality (Cobbe et al., 2021; Lightman et al., 2023). Metareasoning treats the selection of a computation as a decision with a cost and a potential downstream benefit (Russell & Wefald, 1991; Hay et al., 2014). Their connection is operational: a score becomes useful when it changes which computation receives the remaining budget. A localized defect in an otherwise promising trajectory motivates repair; uniformly weak reasoning motivates pruning. The distinction requires both an assessment of the trajectory and an action that uses that assessment.

We introduce CoT^2-Meta, a training-free framework for *budgeted metacognitive control* (Figure 1). Strategy-conditioned generation constructs a reasoning tree, process evaluation characterizes its partial trajectories, and a compact meta-state connects this information to expansion, pruning, repair, stopping, and abstention. The empirical argument follows this chain: controlled policy replacement tests state-dependent action selection; evaluator interventions test its information source; candidate and repair audits identify how decisions affect final answers. Across two backbones, the full policy adds 2.0 macro points over threshold-only control with the inference components fixed. The remaining errors differ substantially by task, making the allocation between discovering and preserving correct candidates central to the analysis. The next section positions this trajectory-control

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

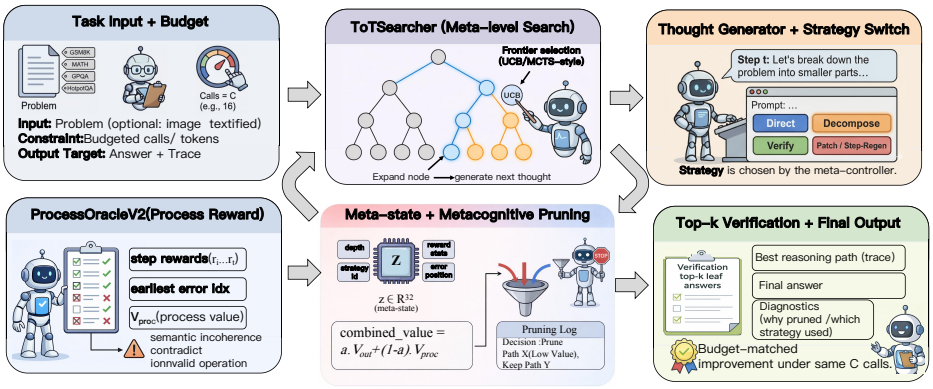


Figure 1: **CoT^2-Meta as reasoning over reasoning.** Object-level reasoning builds partial trajectories; the meta-controller allocates a bounded budget across expand, prune, repair, stop, and abstain actions.

perspective within structured reasoning, process supervision, adaptive computation, and selective prediction.

2 Related Work

Chain-of-thought, self-consistency, and least-to-most prompting organize inference through intermediate reasoning and multiple solution paths (Wei et al., 2022; Wang et al., 2022; Zhou et al., 2022). Tree-of-Thoughts and Graph-of-Thoughts introduce explicit search structures, while ReST-MCTS* couples process-reward-guided search with self-training (Yao et al., 2023; Besta et al., 2024; Zhang et al., 2024); the comparison here uses its inference-only implementation. Compute-aware reasoning further considers how inference effort should be distributed between generation and verification, while adaptive routing selects reasoning behavior under resource constraints (Snell et al., 2024; Singhi et al., 2025; Ma et al., 2026). Complementing these approaches, verifiers and process supervision assess complete solutions and intermediate reasoning steps (Cobbe et al., 2021; Lightman et al., 2023), and Self-Refine and Reflexion use model feedback for iterative revision (Madaan et al., 2023; Shinn et al., 2024). Calibration studies distinguish confidence from empirical correctness, while selective prediction formalizes when a system should withhold an answer (Guo et al., 2017; Jiang et al., 2021; Bartlett & Wegkamp, 2008). CoT^2-Meta brings these directions into a unified trajectory-management framework in which expansion, pruning, local repair, stopping, and abstention are selected according to the current reasoning state. Process signals are therefore not used only to score or rank trajectories, but to determine which computation should be executed next under a finite inference budget. The evaluation correspondingly separates state-conditioned action selection from the underlying generation and evaluation components, examines how process-signal quality affects control decisions, and distinguishes selective emission from additional-budget fallback. This formulation places search, evaluation, repair, and termination within a common budgeted control problem, linking intermediate reasoning quality directly to the allocation of subsequent computation.

3 Budgeted Metacognitive Reasoning

3.1 Reasoning trajectories and the control problem

For an input problem x, a partial trajectory is $\tau_t = (z_1, \ldots, z_t)$, where each z_i is a coherent reasoning unit such as a sub-derivation, decomposition, verification, or intermediate conclusion. The object-level generator proposes a continuation

$$z_{t+1} \sim G_\theta(\cdot \mid x, \tau_t, s_t), \tag{1}$$

where s_t selects Direct, Decompose, or Verify. Continuations become children in a reasoning tree $\mathcal{T}_t$, whose frontier $\mathcal{F}_t$ contains trajectories available for further computation.

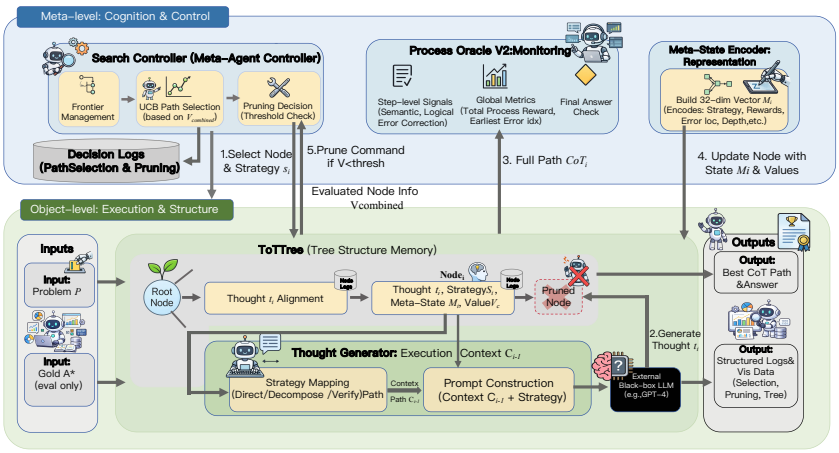


Figure 2: **CoT^2-Meta pipeline.** Strategy-conditioned generation builds a reasoning tree; the online evaluator scores partial trajectories; and the meta-controller converts those signals into meta-states that allocate budget across expand, prune, repair, stop, and abstain. The system returns the best validated trajectory or abstains, together with structured decision traces.

The meta-control policy π selects computations and ultimately returns an answer $\hat{y}$ or abstains. Its cost-constrained formulation is

$$\max_{\pi} \mathbb{E}[U(\hat{y}, x)] \quad \text{subject to} \quad C(\pi; x) \le B, \qquad C(\pi; x) = \sum_{q \in \mathcal{Q}(\pi; x)} \left(T_q^{\text{in}} + T_q^{\text{out}}\right), \tag{2}$$

where $\mathcal{Q}$ includes generation, process evaluation, verification, repair, parsing retries, and every model call triggered by a selected control action. The node–action arbitration itself is deterministic. U measures the terminal task outcome. The experiments match mean total-token expenditure within each task–backbone block and additionally record calls and latency.

3.2 Online process evaluation and the meta-state

Figure 2 separates the trajectory being extended from the evidence used to manage it. The evaluator receives the problem and current trajectory and returns semantic consistency Sem_i, logical consistency Log_i, and self-correction evidence Fix_i, each in $[0, 1]$:

$$r_i = 0.2\,\text{Sem}_i + 0.5\,\text{Log}_i + 0.3\,\text{Fix}_i, \qquad v^{\text{proc}}(\tau_t) = \frac{1}{t}\sum_{i=1}^{t} r_i. \tag{3}$$

A verify-style query provides outcome-oriented confidence $v^{\text{out}}(\tau_t)$. The fused value is

$$v(\tau_t) = \lambda v^{\text{out}}(\tau_t) + (1 - \lambda) v^{\text{proc}}(\tau_t), \qquad \lambda = 0.40. \tag{4}$$

Labeled steps and gold outcomes support the offline validation and counterfactual analyses in Section 4.4; online evaluation uses the problem and generated reasoning.

A deterministic construction $m_t = \phi(x, \tau_t, o_t) \in \mathbb{R}^{32}$ summarizes the active strategy, depth, recent reward history and statistics, process/outcome values, earliest low-health location, repair history, frontier statistics, completion status, and remaining budget. Appendix A.2.1 gives all 32 coordinates, their normalization, and the coordinate-to-action map.

3.3 State-conditioned node–action arbitration

For each frontier node n, the controller first computes the UCB-style frontier score

$$U(n) = v(n) + \beta\sqrt{\frac{\log(V + 1)}{\text{vis}(n) + 1}}, \qquad \beta = 1.25, \tag{5}$$

where V is the number of frontier selections. Let $\bar{U}(n) \in [0,1]$ be the rank-normalized frontier score, $b = (B - C)/B$ the remaining-budget ratio, $R(n)$ the number of previous repairs, and $d(n) = [\theta_{\text{health}} - \min_i r_i]_+ / \theta_{\text{health}}$ the severity of the earliest low-health region, with $\theta_{\text{health}} = 0.50$. The full controller assigns three feasible action utilities

$$\begin{aligned} Q_{\text{exp}}(n) &= \bar{U}(n)\, b\, [1 - d(n)], \\ Q_{\text{rep}}(n) &= \mathbf{1}[v(n) \geq \tau_{\text{prune}}]\, \frac{v(n) d(n) b}{1 + R(n)}, \\ Q_{\text{prune}}(n) &= \mathbf{1}[v(n) < \tau_{\text{prune}}]\, \frac{\tau_{\text{prune}} - v(n)}{\tau_{\text{prune}}} [1 - \bar{U}(n)], \end{aligned} \tag{6}$$

with $\tau_{\text{prune}} = 0.35$. Actions whose next model invocation would exceed the remaining token allocation are removed from the feasible set. The controller then executes

$$(n^\star, a^\star) = \arg \max_{n \in \mathcal{F},\, a \in \{\text{EXPAND}, \text{REPAIR}, \text{PRUNE}\}} Q_a(n). \tag{7}$$

This global comparison allows repair on one trajectory to compete directly with expansion or pruning on another. A completed trajectory with $v(n) \geq \tau_{\text{stop}} = 0.90$ triggers STOP; when the budget is exhausted, the surviving trajectory with largest v^{out} is emitted if $v^{\text{out}} \geq \tau_{\text{abs}} = 0.70$ and otherwise yields ABSTAIN.

For EXPAND, the reasoning mode is also selected from the current state. The three priorities are

$$\begin{aligned} P_{\text{Direct}} &= v(n)[1 - d(n)], & P_{\text{Decompose}} &= [1 - v^{\text{proc}}(n)] b, \\ P_{\text{Verify}} &= \max\{|v^{\text{out}}(n) - v^{\text{proc}}(n)|, \mathbf{1}[n \text{ complete}]\}. \end{aligned} \tag{8}$$

and the largest priority determines the prompt mode, with fixed tie order VERIFY$\succ$DECOMPOSE$\succ$DIRECT. REPAIR identifies $\hat{i} = \min\{i : r_i < \theta_{\text{health}}\}$, preserves the preceding prefix, regenerates the suffix, and re-evaluates the branch.

For policy isolation, threshold-only shares the same $U(n)$, evaluator, thresholds, strategy selector, repair operator, and token ledger, but selects $\arg\max_n U(n)$ before applying STOP$\rightarrow$PRUNE$\rightarrow$REPAIR$\rightarrow$EXPAND to that node. Full CoT^2-Meta instead optimizes feasible node–action pairs across the frontier via Equation 7. Appendix A.2.1 gives both policies in pseudocode.

4 EXPERIMENTS

4.1 TASKS, CONTROLS, AND TOTAL-TOKEN MATCHING

The core tasks are MATH-500 ($N = 500$), GPQA-Diamond ($N = 198$), a fixed BBEH slice ($N = 1{,}000$), and all 880 problems in LiveCodeBench release_v5 (Hendrycks et al., 2021; Rein et al., 2024; Kazemi et al., 2025; Jain et al., 2024). Claude-4.5-Opus and Llama-3.1-8B-Instruct are evaluated at nominal allocations of $8B_{\text{greedy}}$, using the task–backbone reference costs in Table 4. Observed mean expenditure is within $\pm 1\%$ of each nominal allocation across all methods; Appendix B provides the token and runtime ledgers.

The comparisons include sampling, verification, PRM-guided selection, tree search, inference-only ReST-MCTS*, self-feedback, and fixed adaptive routing (Wang et al., 2022; Yao et al., 2023; Zhang et al., 2024; Madaan et al., 2023; Shinn et al., 2024). Each task–backbone comparison shares the active model and answer-extraction interface. Headline inference uses the standalone policy, with fallback disabled and unresolved abstentions scored as incorrect. Results aggregate five seeds. Paired uncertainty uses 10,000 instance-bootstrap resamples and Holm adjustment across the four tasks within each backbone (Appendix H). Mechanism macros cover MATH, GPQA, and BBEH; four-task macros add LiveCodeBench. Appendix B.1 reports the 15-benchmark suite, and Appendix J analyzes the additional original figures.

4.2 CORE-TASK PERFORMANCE AND TOKEN SCALING

Table 1 places the full controller above the strongest task-specific external baseline by 2.8, 4.3, 2.0, and 2.8 points with Claude, and by 3.3, 3.0, 2.8, and 3.2 points with Llama, using the unrounded

Table 1: Core-task results at $8B_{\text{greedy}}$. Bold marks the full controller; the strongest external baseline in each task is underlined.

Method	MATH	GPQA	BBEH	LCB	Macro	Tokens
Claude-4.5-Opus						
Greedy CoT	78.5	74.2	65.4	42.5	65.2	14,105
Best-of-N	84.2	79.6	68.9	47.8	70.1	14,206
Best-of-$(N-1)$ + verifier	86.8	82.2	70.5	49.8	72.3	14,145
PRM-guided Best-of-N	89.4	85.6	73.5	53.4	75.5	14,244
Vanilla ToT	87.4	83.5	71.2	51.2	73.3	14,066
ReST-MCTS*	89.2	85.2	<u>73.8</u>	53.2	75.4	14,278
Self-Refine / Reflexion	85.9	81.4	69.9	49.0	71.6	14,135
Fixed adaptive routing	<u>90.0</u>	<u>86.1</u>	73.4	<u>53.9</u>	75.9	14,198
CoT^2-Meta	**92.8**	**90.4**	**75.8**	**56.7**	**78.9**	14,180
Llama-3.1-8B-Instruct						
Greedy CoT	52.4	45.5	54.8	24.5	44.3	11,938
Best-of-N	58.6	50.4	58.3	28.1	48.9	12,021
Best-of-$(N-1)$ + verifier	59.5	51.2	58.8	28.9	49.6	11,969
PRM-guided Best-of-N	61.0	52.6	59.7	30.0	50.8	12,051
Vanilla ToT	60.2	51.8	59.2	29.4	50.2	11,898
ReST-MCTS*	62.5	54.4	61.0	31.7	52.4	12,081
Self-Refine / Reflexion	58.9	49.9	57.6	28.3	48.7	11,958
Fixed adaptive routing	<u>62.9</u>	<u>54.9</u>	<u>61.3</u>	<u>32.1</u>	52.8	12,012
CoT^2-Meta	**66.2**	**58.0**	**64.1**	**35.3**	**55.9**	12,000

Scores are percentages. Macro is recomputed from the four displayed task scores; tokens are the mean of the four task-level token means, rounded to an integer. LCB denotes LiveCodeBench release_v5. ReST-MCTS* uses inference-only search.

seed totals. The Claude gains remove 28.0% of the baseline's MATH errors and 31.2% of its GPQA errors, compared with 7.6% on BBEH and 6.1% on LiveCodeBench. Thus, comparable absolute gains operate on different residual-error populations. The candidate audit below identifies substantial selection losses on GPQA and a larger proposal deficit on coding.

With Claude, PRM-guided sampling reaches 75.5% macro accuracy and ReST-MCTS* 75.4%, while the full controller reaches 78.9%. Across the extended suite, improvements over ReST-MCTS* range from 1.1 points on GSM8K to 5.2 on GPQA, with textified visual tasks using the shared text interface (Table 8).

With Llama, ReST-MCTS* leads PRM-guided sampling by 1.6 macro points, whereas the two are nearly tied with Claude; the full controller exceeds both. The fixed-component experiment yields similar task-level increments across backbones despite different absolute accuracies, locating the controlled contribution in how available reasoning operations are selected.

At $8\times$, CoT^2-Meta exceeds every included baseline at $16\times$ on all four core tasks (Tables 20–23). Its own $4\times \rightarrow 8\times$ gains are 4.6, 4.4, 2.6, and 4.5 points; the next doubling adds only 1.3, 1.6, 1.3, and 1.5. This combination identifies both an allocation advantage and diminishing returns at the upper operating points. In the call-budget MATH study, Figure 3 likewise separates policy from allocation: at 16 calls, CoT^2-Meta reaches 77.1% versus 74.2% for ReST-MCTS*. Its calibration at that operating point is examined in Figure 4; token-matched calibration is reported in Table 37.

The token-scaling matrices also quantify the changing price of an accuracy gain. On MATH, moving from $4\times$ to $8\times$ spends another 7,720 nominal tokens for 4.6 points, approximately 0.60 points per 1,000 added tokens. Moving from $8\times$ to $16\times$ spends another 15,440 for 1.3 points, approximately 0.084 points per 1,000. The corresponding marginal-return reductions are 5.5-fold on GPQA, 4.0-fold on BBEH, and 6.0-fold on LiveCodeBench. These finite-budget comparisons motivate selective continuation: at the upper allocations, where additional search yields fewer corrected answers per token, identifying a productive intervention becomes increasingly consequential.

The runtime ledgers give a complementary view of allocation (Tables 11 and 12). Relative to fixed routing, the full controller adds 350 input tokens and removes 368 output tokens with Claude; with Llama, the changes are $+250$ and -262. Near-equal total expenditure therefore accompanies a

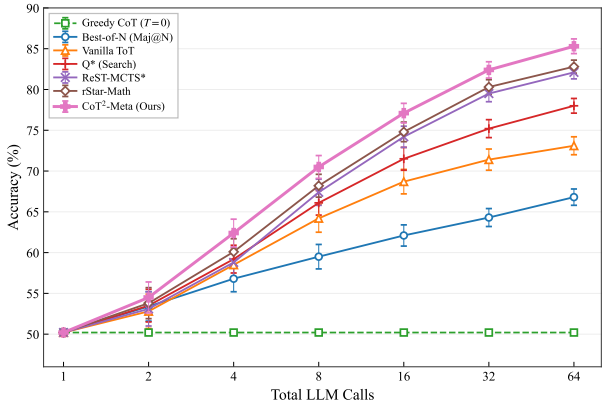


Figure 3: **Compute–accuracy scaling on MATH under a call budget.** Accuracy versus total LLM calls. The horizontal axis counts generation and evaluation queries; the total-token scaling results are reported in Tables 20–23.

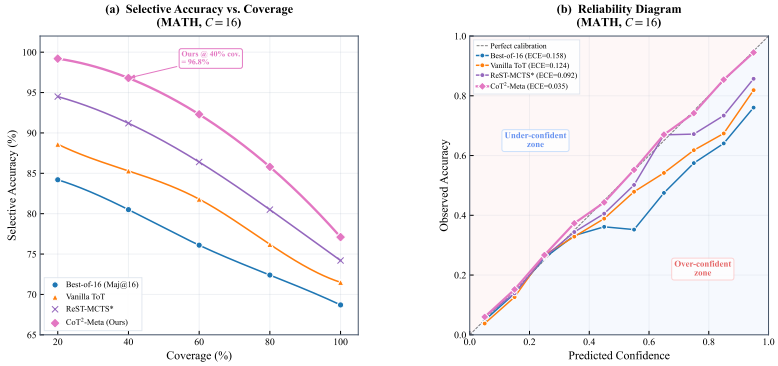


Figure 4: **Calibration and selective prediction on MATH ($C = 16$ calls). (a)** Selective accuracy versus coverage. **(b)** Reliability diagram. These panels use the call-budget evaluation; Table 37 reports calibration at the total-token operating point.

different input/output composition and fewer calls, 13.6 versus 14.2. Relative to ReST-MCTS*, measured latency under parallel candidate execution is 21.2% lower with Claude and 22.0% lower with Llama.

4.3 Isolating the contribution of action selection

Table 2 isolates the arbitration rule defined in Section 3.3. Threshold-only first chooses the highest-U node and applies a fixed action cascade to that node; full CoT^2-Meta scores feasible node–action pairs across the entire frontier with Equation 6. The generator, evaluator, prompts, strategy selector, repair operator, thresholds, and token allocation are shared. The full policy adds 2.0 macro points on each backbone, with task-level gains of 1.9–2.2 points for Claude and 1.8–2.2 for Llama (Table 13). Shuffling $m(n)$ across frontier nodes before action scoring reduces the macros to 73.3% and 50.2%, linking the state–trajectory correspondence directly to the arbitration decision.

The 900-state counterfactual audit relates these gains to decision consequences: each feasible action receives the same remaining allocation and three stochastic continuations. Relative to threshold-only control, agreement rises from 57.8% to 69.1%, normalized regret falls from 0.211 to 0.164 (22.3%), harmful pruning falls by 40.0%, wasted expansion by 29.6%, harmful stopping from 6.9% to 4.2%, and repair precision rises from 68.9% to 80.6% (Table 14). These shifts retain useful paths while spending less search on low-value continuations. At terminal depth, offline best actions remain

Table 2: Action-selection isolation. Task macros cover all four core tasks. The decision audit uses 900 states and three continuations per feasible action.

Policy	Claude	Llama	Agree. ↑	Regret ↓	Prune ↓	Repair ↑
Random feasible	69.2	47.2	29.8	0.342	16.4	44.8
Fixed-stage schedule	71.3	48.9	41.7	0.287	13.1	52.0
Greedy-value	73.4	50.7	49.6	0.253	10.8	60.4
UCB-only	75.2	52.4	53.2	0.228	9.4	63.2
Threshold-only	76.9	53.9	57.8	0.211	8.5	68.9
Shuffled meta-state	73.3	50.2	44.3	0.279	12.7	54.1
Full CoT^2-Meta	78.9	55.9	69.1	0.164	5.1	80.6

Agreement, harmful-prune rate, and repair precision are percentages; audit metrics use the Claude backbone. Generation, evaluation, prompts, action set, candidate interface, repair, and total-token allocation are shared across policies.

heterogeneous—32% stop and 27% repair (Table 15)—so depth alone does not determine the next computation.

4.4 Process-evaluation quality and local repair

The evaluator comparisons connect diagnostic quality to actionable information. The self-evaluator obtains AUROC 0.881 on 3,200 held-out PRM800K steps and 0.846 on 1,200 manually labeled cross-domain steps (Appendix D). In Table 3, changing to GPT-4o alters final task scores by at most 0.3 points, and changing to Qwen2.5-72B by at most 0.9. Removing signal components has larger effects: logical-only evaluation trails the full signal by 2.0–2.6 points, and semantic-only evaluation by 3.9–4.9. These contrasts support combining complementary assessments of the trajectory; evaluator identity alone produces a smaller variation in this comparison.

Controlled degradation reveals how signal use can fail. At Gaussian noise $\sigma = 0.50$, macro accuracy falls from 86.3% to 80.7%, with GPQA losing 6.5 points. A constant signal yields 78.1%, compared with 77.3% for permutation and 76.7% for random scores (Table 19). Near-chance discrimination can therefore produce different outcomes: score values and their assignment to trajectories also affect the controller's threshold decisions. The offline gold-informed diagnostic reaches 94.2%, 92.0%, and 77.5%, locating additional opportunity in the quality of the control signal.

The localization diagnostics identify a second requirement beyond ranking trajectories. Relative to logical-only scoring, the composite self-evaluator raises the earliest-error score from 0.654 to 0.711 and repair precision from 0.732 to 0.785 (Table 17). Gaussian perturbation tests score discrimination, while the localization audit tests where repair is placed; together they separate mis-scored trajectories from mislocalized interventions.

Local repair tests the computation performed after an error is identified. Selective repair and full regeneration have similar trigger rates, 32.4% and 31.8%, but produce 92.8% and 89.4% MATH accuracy at matched token expenditure. Selective repair preserves 94.2% of prefixes and has a 6.8% hurt rate, versus 0.0% and 18.9% for regeneration (Table 28). The comparison links successful intervention to retaining useful reasoning and replacing its defective suffix. Exact localization is associated with 68% success and 3% harm; at distance at least three, these become 16% and 26% (Table 29). Appendix F analyzes the trajectory-level rates alongside example-level accuracy.

The repair-all comparison further separates the availability of a repair operator from selective use of it. Repair-all uses 15,488 tokens and reaches 90.8%, compared with 15,440 tokens and 92.8% for selective repair. Its cost per rescued trajectory is 980 tokens versus 610, a 37.8% reduction for the selective policy. Extra expansion reaches 90.1% at 15,452 tokens. These contrasts identify the joint value of choosing when to revise and how much reasoning to preserve: the same overall allocation buys more successful correction when repair is directed toward suitable trajectories.

Component removal places these operations within the full loop: removing tree interaction costs 7.0 macro points, process scoring 4.4, and selective repair 2.9, with unused compute reassigned to generation (Table 16). Figure 5 illustrates the retention problem on the hard subset: pruning

Table 3: Process-evaluator replacement with fixed generator and controller.

Evaluator	AUROC	AUPRC	F1	MATH	GPQA	BBEH
Self-evaluator	0.846	0.784	0.742	92.8	90.4	75.8
GPT-4o judge	0.858	0.798	0.755	93.1	90.6	76.0
Qwen2.5-72B judge	0.823	0.751	0.714	92.1	89.5	74.9
Semantic-only	0.742	0.667	0.641	88.9	85.5	71.9
Logical-only	0.807	0.735	0.701	90.8	87.8	73.7
Random evaluator	0.501	0.425	0.431	83.4	79.0	67.8
Offline gold evaluator	0.952	0.927	0.891	94.2	92.0	77.5

Step metrics use the 1,200-step cross-domain manual set. The final row is a separate offline, label-informed diagnostic. Online evaluator inputs are the problem and the current trajectory.

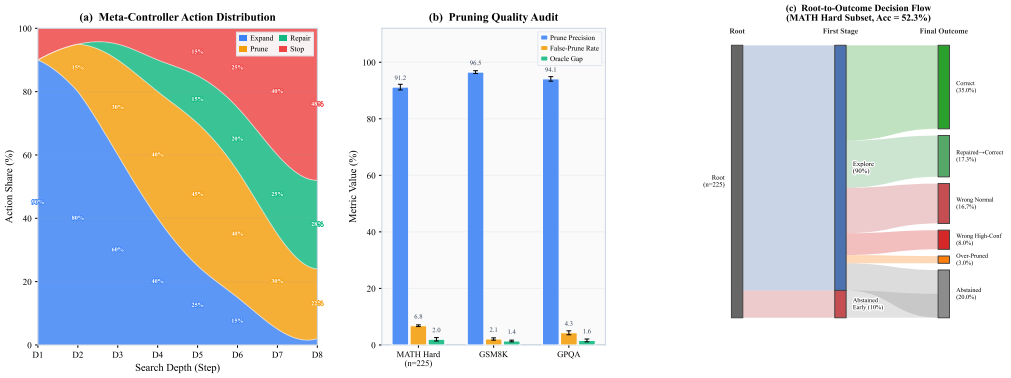


Figure 5: **Decision trace interpretability of CoT^2-Meta** on the curated hard subset of MATH. **(a)** Action distribution over search depth, showing early expansion, mid-stage pruning, and late repair/stop behavior. **(b)** Pruning audit across datasets, comparing prune precision, false-prune rate, and oracle gap. **(c)** Root-to-outcome flow on the MATH hard subset.

precision is 91.2%, yet the false-prune rate is 6.8%, versus 2.1% on GSM8K. Efficient pruning and preservation of recoverable paths are distinct objectives that the process signal must balance.

4.5 Candidate Coverage and Residual Errors

Table 24 decomposes the three-task performance difference through candidate availability and final selection. Relative to ReST-MCTS*, the 3.6-point accuracy advantage comprises a 2.0-point coverage increase and a 1.6-point reduction in the coverage–accuracy gap; generation, evaluation, and control jointly determine both quantities. At $2\times$, CoT^2-Meta has slightly lower coverage, 85.5% versus 86.0%, but higher accuracy, 78.5% versus 76.9% (Table 26); better conversion of available solutions already matters at low budgets.

The designated-run counts expose task-specific priorities (Table 25). On GPQA, 15 of 19 errors occur despite a correct candidate being generated; on LiveCodeBench, 264 of 381 errors occur without one. Across 2,578 examples, the 678 errors comprise 428 proposal misses and 250 failures after candidate availability. Evaluator misranking and controller errors account for 161 of these 250 cases (64.4%), concentrating downstream losses in assessment and action selection. BBEH and LiveCodeBench contribute 409 of those 428 misses. The aggregate predominance of proposal failure thus reflects a concentrated task-level deficit, while scientific QA places greater pressure on selection and preservation. Appendix E.2 connects these counts to evaluator, controller, execution, and budget attributions.

For a fixed task in this audit, let q be the fraction of problems with a correct generated candidate and s the fraction of those problems answered correctly. Final accuracy satisfies

$$a = q\,s, \qquad 1 - a = (1 - q) + q(1 - s). \tag{9}$$

The first error term is proposal absence; the second is loss after availability. The designated-run conversion rates s are 95.7% on MATH, 92.3% on GPQA, 88.7% on BBEH, and 81.0% on LiveCodeBench. Coding therefore combines the lowest coverage with the weakest conversion of available solutions. Scientific QA instead has near-complete coverage and a concentrated selection problem. These task-specific factors explain why a shared controller needs both proposal-changing actions and actions that preserve, repair, or select existing reasoning.

Restoring strategy switching raises effective branches from 8.6 to 12.8, reduces duplication from 25.2% to 11.2%, increases coverage by 1.8 points, and narrows the selection gap by 0.9 points (Table 24), together yielding a 2.7-point gain over no switching. The larger effective-branch count and coverage indicate more varied candidate pools with more correct opportunities, while the smaller selection gap shows that more of those opportunities survive final selection. Local repair complements this by preserving useful reasoning within a selected trajectory.

4.6 Selective prediction and threshold transfer

Confidence is useful when it identifies where withholding or further computation changes the outcome. In the fixed MATH ranking, the top 442 predictions contain 439 correct answers. Withholding the remaining 58 removes 33 of 36 errors while retaining 94.6% of the correct answers, giving 99.3% selective accuracy at 88.4% coverage (Table 38). The same prefix contributes 87.8% correct answers over all 500 problems. At matched coverage, fixed routing retains eight errors versus three for CoT^2-Meta. ECE falls from 0.061 to 0.030 and Brier score from 0.094 to 0.072 (Table 37). Figure 4 provides the call-budget counterpart: the confidence ordering and its probability calibration assess different aspects of the emission decision.

Freezing a global cross-model threshold configuration costs 0.7 macro points relative to per-task tuning, compared with the 5.6-point loss from shuffled states with Claude (Tables 31 and 2). Within these interventions, matching actions to trajectory information matters more than local threshold specialization.

Fusion and pruning are coupled because changing λ shifts branch value by $(v^{\mathrm{out}} - v^{\mathrm{proc}})\Delta\lambda$, after which the pruning threshold determines retention. In the joint sweep, the central 3×3 neighborhood spans 85.8–86.3% macro accuracy, while the outer corners fall to 84.9–85.0% (Table 33). Together with threshold transfer, this identifies a broad interior operating region and larger losses when evidence fusion and retention move jointly toward the extremes.

Additional-budget fallback yields six and nine net extra correct MATH answers at 17.8% and 34.1% more mean tokens, respectively (Table 39). The lowest-confidence 40 predictions contain 32 errors, concentrating candidates for intervention, but broader rerouting has lower marginal return: standalone-to-low-threshold routing costs about 2,292 additional mean tokens per accuracy point, versus 4,200 from low to high threshold. Confidence therefore identifies where errors concentrate, while realized benefit still depends on subsequent search. Appendix I analyzes these operating decisions and the 91-problem LiveCodeBench-Plus pilot.

5 Discussion and Conclusion

CoT^2-Meta makes trajectory state an operational input to test-time reasoning. The evidence links three stages: informative process scores distinguish useful from defective reasoning; state-dependent actions allocate search and repair; final selection converts available candidates into answers. Policy replacement, score perturbation, and token-matched repair comparisons test different parts of this chain. Together, they support the central role of choosing the appropriate computation for the current trajectory.

The error structure also gives this design a task-dependent emphasis. GPQA frequently contains a correct candidate that is lost before output, whereas most LiveCodeBench failures lack one in the evaluated search. Preserving and selecting available solutions is therefore a different computational objective from discovering additional solutions. Confidence-prefix and fallback results extend the same principle to termination: computation has value where a further intervention can improve the decision. The resulting inference framework couples proposal diversity, localized correction, and selective emission to the trajectory evidence that determines their usefulness at the current allocation.

AI USE STATEMENT

Generative AI tools were used solely for language polishing of the manuscript.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Peter L Bartlett and Marten H Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(8), 2008.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 9, 2021.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Nicholas Hay, Stuart Russell, David Tolpin, and Solomon Eyal Shimony. Selecting computations: Theory and applications. *arXiv preprint arXiv:1408.2048*, 2014.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

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

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K Jain, Virginia Aglietti, Disha Jindal, Yuanzhu Peter Chen, et al. Big-bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26473–26501, 2025.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 3214–3252, 2022.

Siyuan Ma, Bo Gao, Xiaojun Jia, Simeng Qin, Tianlin Li, Ke Ma, Xiaoshuang Jia, Wenqi Ren, and Yang Liu. Odar: Principled adaptive routing for llm reasoning via active inference. *arXiv preprint arXiv:2602.23681*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback, 2023. *URL https://arxiv. org/abs/2303.17651*, 2303, 2023.

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

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First conference on language modeling*, 2024.

Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial intelligence*, 49(1-3):361–395, 1991.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. *URL https://arxiv. org/abs/2303.11366*, 8, 2024.

Nishad Singhi, Hritik Bansal, Arian Hosseini, Aditya Grover, Kai-Wei Chang, Marcus Rohrbach, and Anna Rohrbach. When to solve, when to verify: Compute-optimal problem solving and generative verification for llm reasoning. *arXiv preprint arXiv:2504.01005*, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, et al. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15134–15186, 2025.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772, 2024.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models, 2023. *URL https://arxiv. org/abs/2205.10625*, 2022.

A Evaluation Protocol and Practical Instantiation

The inference interface links trajectory construction to a sequence of resource-consuming decisions. Its evaluation therefore records both what a method returns and how it distributes computation among generation, evaluation, verification, and repair. The protocol below defines the units used in the mechanism analyses.

A.1 Core datasets and reference allocations

Table 4: Dataset sizes and nominal total-token allocations.

Dataset	N	Claude		Llama	
		B_g	$8B_g$	B_g	$8B_g$
MATH-500	500	1,930	15,440	1,600	12,800
GPQA-Diamond	198	1,540	12,320	1,300	10,400
BBEH fixed slice	1000	1,340	10,720	1,200	9,600
LiveCodeBench release_v5	880	2,280	18,240	1,900	15,200

$B_g = B_{\text{greedy}}$. MATH uses exact match, GPQA accuracy, BBEH macro accuracy, and LiveCodeBench Pass@1.

Table 4 defines a task-relative allocation: the $8B_{\text{greedy}}$ multiplier corresponds to 10,720 tokens on Claude/BBEH and 18,240 on Claude/LiveCodeBench, a 1.70-fold difference in absolute expenditure. The corresponding Llama allocations are 9,600 and 15,200 tokens. Normalizing to a task–backbone reference preserves these differences in the cost of producing a baseline solution. The controller then decides how to divide that allocation among reasoning operations. Consequently, an $8\times$ comparison concerns alternative allocations within a task; the four-task mean in Table 1 summarizes those comparisons. The scaling experiment uses $2B_{\text{greedy}}$, $4B_{\text{greedy}}$, $8B_{\text{greedy}}$, and $16B_{\text{greedy}}$ with Claude, with matching assessed from measured mean input-plus-output tokens.

MATH-500 uses exact answer matching, GPQA-Diamond the canonical multiple-choice answer space, BBEH a fixed 1,000-example slice with macro accuracy, and LiveCodeBench code-execution Pass@1. Three-task mechanism macros give equal weight to MATH, GPQA, and BBEH; four-task macros include LiveCodeBench. The 15-task mean in Table 8 averages task-standard metrics. These task-weighted summaries complement the example-weighted error counts in Table 27, which expose how the larger BBEH and coding populations contribute to residual failures.

Table 5: Frozen LiveCodeBench evaluation specification.

Field	Specification
Dataset	`livecodebench/code_generation_lite`; code-generation scenario
Snapshot freeze	2026-09-18 UTC
Release selector	`release_v5`
Release interval	2023-05-01 through 2025-01-31, inclusive
Sample definition	All 880 `question_id` values in `release_v5:test`, in dataset-native order
Manifest key	`livecodebench/code_generation_lite: release_v5:test:all-880`

The frozen release in Table 5 supplies the common 880-problem population for coding accuracy, candidate availability, and residual-error attribution. Their connection is important for the core argument: finding a correct program somewhere in a search and returning a program that passes the tests are successive events on the same problems. The gap between 616 examples with a correct candidate and 499 correct outputs in Table 25 therefore measures a loss after candidate availability within this release. The 91-problem LiveCodeBench-Plus pilot defines a different evaluation population and is analyzed in Appendix I.

Answer extraction and scoring are shared across methods. Math extraction uses boxed answers or explicit final-answer markers, followed by the final numerical expression when those markers are

absent. Multiple-choice outputs map option letters or text spans to the canonical label. Textified visual tasks use the shared OCR-and-caption representation supplied by the experimental pipeline.

A.2 Generation, evaluation, and state construction

The generator uses temperature 0.4, top-p 0.95, and a maximum output length of 2,048 tokens per query. The structured evaluator and terminal verifier use temperature 0.0 and a maximum output length of 512 tokens; the greedy baseline uses temperature 0.0. Query limits govern individual outputs, while the cost ledger includes input tokens and the sequence of invoked queries.

The three generation modes act through the same backbone. DIRECT extends the derivation, DECOMPOSE introduces intermediate subgoals, and VERIFY inspects the current reasoning and supports a confidence judgment. Each generated thought defines one reasoning step. Nodes retain parent links, depth, strategy tags, quality values, and control metadata.

A.2.1 Exact 32-dimensional state

The state is constructed after every evaluation and before every action decision. Recent rewards are left-padded with zeros when fewer than ten steps exist and truncated to the ten most recent values thereafter. The normalized UCB rank is $\bar{U}(n) = (\mathrm{rank}_{\mathcal{F}} U(n) - 1)/\max(1, |\mathcal{F}| - 1)$, with rank zero assigned to the lowest frontier score. Repair deltas are zero before the first repair.

Table 6: Exact coordinates of the 32-dimensional online meta-state and their direct control uses.

Coords.	Definition	Direct use
1–3	one-hot active mode: Direct, Decompose, Verify	strategy history
4	$t/(t+1)$	strategy priority
5–14	ten most recent r_i values	defect/reward history
15–19	mean, std., min, max, and latest r_i	strategy, repair
20–22	v^{proc}, v^{out}, and fused v	all value decisions
23	$\hat{i}/t$ if a low-health step exists, else 0	repair location
24	$t^{-1}\sum_i \mathbf{1}[r_i < \theta_{\mathrm{health}}]$	expansion/repair
25	indicator that the trajectory has been repaired	repair history
26	$R/(R+1)$, where R is repair count	repeated-repair penalty
27–28	last repair deltas Δv, Δv^{proc}	repair history
29	$\mathrm{vis}(n)/(\mathrm{vis}(n)+1)$	frontier exploration
30	rank-normalized $U(n)$	expansion/pruning
31	completed-trajectory indicator	verify/stop
32	$(B-C)/B$	action and strategy allocation

Equation 6 reads coordinates 17 and 20–32 through $d(n)$, $v(n)$, repair count, frontier rank, and remaining budget. Equation 8 reads coordinates 20–24, 31, and 32. The stop decision reads 22 and 31; terminal abstention reads 21 and 32. This produces a deterministic mapping from the state vector to every online control decision.

Table 7: Process-evaluation signals.

Signal	Meaning
Semantic	Semantic alignment with the problem and current reasoning goal.
Logical	Logical coherence and inferential validity of the current step.
Fix	Evidence that the step explicitly detects, revises, rejects, or corrects a previous incorrect intermediate claim.

The signals in Table 7 address different properties of a continuation. Semantic alignment concerns whether it serves the current problem; logical validity concerns whether its inferences hold; the Fix score records explicit correction of an earlier claim. Their weighted combination, $(0.2, 0.5, 0.3)$, makes logical validity the largest contribution to a step reward while retaining relevance and revision information. Table 3 supports the composite signal through its advantage over semantic-only and logical-only scoring. The measured contrast concerns the combined signal, including its interaction with control.

Trajectory averaging and earliest-error localization use that signal at different resolutions. A single low reward changes a length-t mean in proportion to $1/t$, whereas the earliest-error coordinate preserves its position as a repair target. This distinction explains why a useful overall prefix can coexist with a local intervention point. The controller's fusion weight of 0.40 combines trajectory-level process evidence with terminal confidence; the localization audit tests the separate question of whether the identified point is the right place to revise.

A.3 Structured scoring and parsing

The process-scoring prompt has the following interface:

> **Role:** You are a reasoning-process evaluator.
> **Inputs:** A problem and a multi-step reasoning trajectory.
> **Instruction:** Evaluate each reasoning step independently.
> **Returned scores:** Semantic alignment with the problem and current reasoning goal; logical coherence and inferential validity; and whether the step explicitly detects, revises, rejects, or corrects a previous incorrect intermediate claim.
> **Output format:**
> `Step 1: Semantic=0.xx, Logical=0.xx, Fix=0.xx`
> `Step 2: Semantic=0.xx, Logical=0.xx, Fix=0.xx`
> The format continues for every controller-defined step.
> **Constraint:** Return only the step-wise scores. Do not provide explanations.
> **Problem slot:** `{problem}`
> **Trajectory slot:** `{trajectory}`

A parse failure occurs when a required field is missing, non-numeric, or outside $[0, 1]$, or when the number of returned steps differs from the trajectory segmentation. The controller issues one constrained re-prompt requesting the same step-indexed numeric format. Fields that remain invalid after this retry receive value zero. Verify-style confidence uses the same retry-and-default rule, with original queries and retries both charged to the ledger.

The zero default has an operational consequence: an invalid score contributes low process evidence rather than an unparsed string to frontier selection and repair localization. Re-prompting seeks usable evidence before that default is applied. The intervention tables vary evaluator information after this structured interface, while the noisy-format profile in Figure 7 describes the end-to-end response to formatting perturbations.

The repair prompt receives the trajectory, the detected error position, and the preserved prefix. It requests suffix regeneration beginning at that position; the resulting branch is scored again before subsequent control decisions. Re-evaluation closes the loop between detecting a defect, acting on it, and assessing the result. A repair attempt thus creates new evidence for allocation as well as new reasoning content.

A.4 Inference workflow and accounting

The full controller compares all feasible node–action pairs before execution. Threshold-only selects one frontier node first and then applies its fixed priority cascade. Both controllers use the same strategy selector: DIRECT is favored by high fused value and low defect severity, DECOMPOSE by low process value while budget remains, and VERIFY by process–outcome disagreement or trajectory completion. In the shuffled-state condition, the complete $m(n)$ vectors are permuted across frontier nodes immediately before Equations 6 and 8 are evaluated; the tree, generated text, and token ledger remain unchanged at that decision step.

The shared defaults for fusion, exploration, pruning, repair health, stopping, and abstention are 0.40, 1.25, 0.35, 0.50, 0.90, and 0.70, respectively. Appendix G analyzes development selection and transfer. The ledger sums input and output usage for every model invocation, including re-evaluation and retries. Deterministic bookkeeping constructs the state, ranks frontier nodes, and computes action utilities; every triggered model invocation incurs measured token usage. Standalone inference ends with its answer or abstention. Hybrid rerouting adds the computation reported in Table 39.

Algorithm 1 Full CoT^2-Meta controller

Require: Problem x, backbone G_θ, token allocation B

Initialize root, tree $\mathcal{T}$, frontier $\mathcal{F}$, token ledger $C \leftarrow 0$
while $\mathcal{F} \neq \varnothing$ and $C < B$ **do**
  Evaluate/update $m(n), v(n), U(n)$ for every $n \in \mathcal{F}$
  if $\exists$ completed n with $v(n) \geq \tau_{\text{stop}}$ **then**
    return answer of $\arg\max v(n)$ among those nodes
  end if
  Compute b, $d(n)$, $\bar{U}(n)$ and feasible $Q_{\text{exp}}, Q_{\text{rep}}, Q_{\text{prune}}$
  $(n^\star, a^\star) \leftarrow \arg\max_{n,a} Q_a(n)$
  if $a^\star = \text{EXPAND}$ **then**
    $s^\star \leftarrow \arg\max\{P_{\text{Direct}}, P_{\text{Decompose}}, P_{\text{Verify}}\}$
    Generate one child under $s^\star$; evaluate it; update $\mathcal{T}, \mathcal{F}, C$
  else if $a^\star = \text{REPAIR}$ **then**
    Locate $\hat{i}$; preserve prefix $1{:}\hat{i}-1$; regenerate suffix
    Re-evaluate repaired branch; update repair history, $\mathcal{F}$, and C
  else if $a^\star = \text{PRUNE}$ **then**
    Remove $n^\star$ from $\mathcal{F}$
  end if
end while
$n^\star \leftarrow \arg\max_{n \in \mathcal{F}} v^{\text{out}}(n)$ among surviving completed candidates
return answer of $n^\star$ if $v^{\text{out}}(n^\star) \geq \tau_{\text{abs}}$; otherwise **return** ABSTAIN

Algorithm 2 Threshold-only controller in the policy-isolation study

Require: Same generator, evaluator, strategy selector, repair operator, thresholds, and B as Algorithm 1

while $\mathcal{F} \neq \varnothing$ and $C < B$ **do**
  $n^\star \leftarrow \arg\max_{n \in \mathcal{F}} U(n)$
  if $n^\star$ complete and $v(n^\star) \geq \tau_{\text{stop}}$ **then**
    return answer of $n^\star$
  else if $v(n^\star) < \tau_{\text{prune}}$ **then**
    prune $n^\star$
  else if $\min_i r_i < \theta_{\text{health}}$ **then**
    repair $n^\star$ from its earliest low-health step
  else
    select $s^\star$ with Equation 8 and expand $n^\star$
  end if
end while
Apply the same terminal answer/abstention rule as Algorithm 1

B Extended Results and Cost Accounting

B.1 The 15-benchmark evaluation

Table 8: Extended 15-benchmark results with Claude-4.5-Opus.

Dataset	Greedy	BoN	ToT	ReST	CoT^2-Meta
MATH-500 (Hendrycks et al., 2021)	78.5	84.2	87.4	89.2	92.8
GSM8K (Cobbe et al., 2021)	94.2	96.1	96.8	97.5	98.6
GPQA-Diamond (Rein et al., 2024)	74.2	79.6	83.5	85.2	90.4
MMLU-Pro (Wang et al., 2024)	78.5	82.2	84.5	86.1	88.4
TruthfulQA (Lin et al., 2022)	81.5	85.4	88.2	90.2	92.8
Natural Questions (Kwiatkowski et al., 2019)	36.5	39.8	41.5	43.2	45.4
MMMU-Pro (textified) (Yue et al., 2025)	68.4	73.1	77.8	81.3	85.6
HLE (textified) (Phan et al., 2025)	32.5	38.4	42.1	44.5	48.8
BBEH (Kazemi et al., 2025)	65.4	68.9	71.2	73.8	75.8
ARC-Challenge (Clark et al., 2018)	76.2	80.1	82.4	84.2	86.3
BBH Unseen (Suzgun et al., 2023)	74.5	78.2	80.5	82.1	84.5
HotpotQA (Yang et al., 2018)	78.5	82.5	85.1	87.4	90.4
HumanEval (Chen et al., 2021)	60.5	65.4	68.2	70.4	72.8
MBPP (Austin et al., 2021)	68.5	72.5	75.1	76.8	79.5
LiveCodeBench (Jain et al., 2024)	42.5	47.8	51.2	53.2	56.7
Unweighted mean	67.4	71.6	74.4	76.3	79.3

The suite uses the $8B_{\text{greedy}}$ protocol. Scores follow each task's evaluation metric; this heterogeneous-task mean is distinct from the four-task macro in Table 1. Visual inputs are supplied through the shared textification interface. ReST denotes ReST-MCTS*.

The extended task matrix in Table 8 shows where additional control has different leverage. On GSM8K, the gain over ReST-MCTS* is 1.1 points, from 97.5% to 98.6%; the remaining error fraction falls from 2.5% to 1.4%, a 44% relative reduction. On GPQA, the 5.2-point gain operates at a lower baseline accuracy and removes approximately 35% of its errors. These two cases illustrate why the smallest absolute margin can still be consequential near a high-accuracy operating point. The four-task main comparison includes additional verifier, PRM, feedback, and routing baselines; its task-specific contrasts use those stronger alternatives.

The gains over ReST-MCTS* are 3.0 points on HotpotQA, 2.4 on HumanEval, 2.7 on MBPP, and 3.5 on LiveCodeBench. The same control interface therefore accompanies improvements in multi-hop answer construction and executable program output, despite their different scoring rules. Natural Questions remains at 45.4% after a 2.2-point improvement, locating a substantial unresolved population even with control. For textified MMMU-Pro and HLE, the 4.3-point improvements assess reasoning over the supplied OCR-and-caption representation. Together, the rows establish breadth across output spaces; the policy, evaluator, and repair interventions provide the more specific evidence about how the controller uses trajectory information.

B.2 Task-level token matching

Table 9 ties the Claude performance comparisons to their task-level expenditure. Relative to fixed routing, CoT^2-Meta uses 23 fewer tokens on MATH, 18 fewer on GPQA, 17 fewer on BBEH, and 15 fewer on LiveCodeBench. Each difference is below 0.2% of the corresponding allocation, while the accuracy gains are 2.8, 4.3, 2.4, and 2.8 points against that particular baseline. ReST-MCTS* uses more tokens in all four rows and is the strongest external comparator on BBEH. This expenditure pattern directs the interpretation toward how computation is used within a task. The underlying allocation differences are quantified by the interventions in Table 2, where the inference components are held fixed.

The Llama ledger in Table 10 reproduces the same within-backbone pattern at lower reference allocations. Fixed routing exceeds the full controller by only 15, 12, 10, and 12 tokens across the four tasks; ReST-MCTS* exceeds it by 90, 75, 70, and 90. Llama's 3.1-point displayed macro advantage over fixed routing is therefore obtained at almost identical mean expenditure. The largest deviation

Table 9: Dataset-level total-token accounting: Claude.

Method	MATH	GPQA	BBEH	LCB	Max. dev.
Greedy CoT	15,362	12,261	10,660	18,138	0.56%
Best-of-N	15,472	12,343	10,745	18,262	0.23%
Best-of-$(N-1)$ + verifier	15,408	12,296	10,692	18,184	0.31%
PRM-guided Best-of-N	15,510	12,384	10,772	18,310	0.52%
Vanilla ToT	15,305	12,214	10,635	18,110	0.87%
ReST-MCTS*	15,555	12,405	10,802	18,351	0.76%
Self-Refine / Reflexion	15,400	12,278	10,680	18,180	0.37%
Fixed adaptive routing	15,463	12,338	10,737	18,255	0.16%
CoT^2-Meta	15,440	12,320	10,720	18,240	0.00%

Entries are reported task-level means. Maximum deviation is computed against the corresponding nominal allocation in Table 4.

Table 10: Dataset-level total-token accounting: Llama.

Method	MATH	GPQA	BBEH	LCB	Max. dev.
Greedy CoT	12,736	10,350	9,544	15,120	0.58%
Best-of-N	12,820	10,422	9,622	15,218	0.23%
Best-of-$(N-1)$ + verifier	12,772	10,374	9,570	15,160	0.31%
PRM-guided Best-of-N	12,855	10,446	9,645	15,258	0.47%
Vanilla ToT	12,690	10,310	9,510	15,080	0.94%
ReST-MCTS*	12,890	10,475	9,670	15,290	0.73%
Self-Refine / Reflexion	12,760	10,362	9,560	15,150	0.42%
Fixed adaptive routing	12,815	10,412	9,610	15,212	0.12%
CoT^2-Meta	12,800	10,400	9,600	15,200	0.00%

Entries are reported task-level means. Maximum deviation is computed against the corresponding nominal allocation in Table 4.

anywhere in this ledger is 0.94%, for ToT on BBEH. Each ledger concerns matching inside its backbone; different models have different token reference costs and absolute capabilities. The four-task means, 14,180 tokens with Claude and 12,000 with Llama, summarize these task-specific operating points.

B.3 Aggregate query and runtime profiles

Table 11: Aggregate token and runtime profile: Claude.

Method	Input	Output	Total	Calls	Latency (s)
Greedy CoT	8,120	5,985	14,105	8.0	41.2
Best-of-N	7,600	6,606	14,206	16.0	47.8
Best-of-$(N-1)$ + verifier	7,900	6,245	14,145	16.0	50.5
PRM-guided Best-of-N	8,500	5,744	14,244	15.8	55.1
Vanilla ToT	8,200	5,866	14,066	15.1	53.3
ReST-MCTS*	9,000	5,278	14,278	15.9	61.8
Self-Refine / Reflexion	8,000	6,135	14,135	14.8	52.0
Fixed adaptive routing	8,300	5,898	14,198	14.2	49.4
CoT^2-Meta	8,650	5,530	14,180	13.6	48.7

The aggregation covers the four core tasks. Latency is the reported elapsed time with parallel candidate execution; total-token expenditure is the matching variable and call counts are measured outcomes.

Table 11 reveals a change in the composition of computation. Against fixed routing, the full controller uses 350 more input tokens but 368 fewer output tokens, leaving total expenditure nearly unchanged. The input share is 61.0%, compared with 58.5% for fixed routing; this is the measured input/output mix across all operations, including generation and evaluation. Average calls fall from 14.2 to 13.6, so the ratio of mean tokens to mean calls increases from about 1,000 to 1,043.

A smaller number of calls can therefore carry more context per invocation while allocating fewer tokens to generated output.

Against ReST-MCTS*, the full controller uses 14.5% fewer calls and its measured latency is 21.2% lower, 48.7 versus 61.8 seconds. Best-of-N is slightly faster at 47.8 seconds but has a macro score of 70.1%, compared with 78.9% for CoT^2-Meta. These implementation-level measurements, with parallel candidate execution, show why token expenditure, query count, elapsed time, and answer quality are complementary quantities: the best allocation is determined by the useful reasoning produced, as well as its execution cost.

Table 12: Aggregate token and runtime profile: Llama.

Method	Input	Output	Total	Calls	Latency (s)
Greedy CoT	6,800	5,138	11,938	8.0	17.1
Best-of-N	6,450	5,571	12,021	16.0	20.4
Best-of-$(N-1)$ + verifier	6,700	5,269	11,969	16.0	21.3
PRM-guided Best-of-N	7,200	4,851	12,051	15.8	23.0
Vanilla ToT	7,000	4,898	11,898	15.1	22.1
ReST-MCTS*	7,600	4,481	12,081	15.9	25.4
Self-Refine / Reflexion	6,800	5,158	11,958	14.8	21.4
Fixed adaptive routing	7,100	4,912	12,012	14.2	20.2
CoT^2-Meta	7,350	4,650	12,000	13.6	19.8

The aggregation covers the four core tasks. Latency is the reported elapsed time with parallel candidate execution; total-token expenditure is the matching variable and call counts are measured outcomes.

Table 12 provides the corresponding decomposition with Llama. Moving from fixed routing to the full controller adds 250 input tokens and removes 262 output tokens; the input share changes from 59.1% to 61.3%. Latency falls modestly from 20.2 to 19.8 seconds, while macro accuracy rises from 52.8% to 55.9%. Relative to ReST-MCTS*, latency falls by 22.0%, from 25.4 to 19.8 seconds, at almost equal token expenditure. The similar cost composition on both backbones accompanies the same 2.0-point policy-isolation gain. The runtime tables describe these evaluated implementations; the controlled policy comparison attributes the accuracy contrast to their use of the shared inference operations.

C Controller-Policy and Component Analyses

C.1 Action-selection isolation

Table 13: Task-level controller-policy comparisons.

Policy	MATH	GPQA	BBEH	LCB	Macro
Claude					
Random feasible	83.0	78.8	67.4	47.4	69.2
Fixed-stage schedule	85.5	81.4	69.2	49.0	71.3
Greedy-value	87.4	84.1	70.8	51.2	73.4
UCB-only	89.1	86.2	72.5	53.1	75.2
Threshold-only	90.9	88.2	73.9	54.7	76.9
Shuffled meta-state	87.0	84.2	70.3	51.6	73.3
Full CoT^2-Meta	92.8	90.4	75.8	56.7	78.9
Llama					
Random feasible	56.5	49.0	56.2	27.0	47.2
Fixed-stage schedule	58.3	50.8	57.8	28.8	48.9
Greedy-value	60.4	52.7	59.4	30.4	50.7
UCB-only	62.1	54.6	61.0	32.0	52.4
Threshold-only	64.2	56.0	62.3	33.1	53.9
Shuffled meta-state	59.8	52.0	58.9	29.9	50.2
Full CoT^2-Meta	66.2	58.0	64.1	35.3	55.9

The underlying inference components are held fixed within each backbone.

Table 13 shows a consistent ordering of information use across tasks. With Claude, a fixed-stage schedule reaches 71.3% macro accuracy, greedy-value selection 73.4%, UCB-only control 75.2%, threshold-only control 76.9%, and the full policy 78.9%. With Llama, the corresponding sequence is 48.9%, 50.7%, 52.4%, 53.9%, and 55.9%. The full-versus-threshold contrasts are particularly informative: the improvements cover all four tasks and vary only from 1.9 to 2.2 points with Claude and from 1.8 to 2.2 with Llama. The residual benefit of full action selection is thus distributed across the task families in this controlled comparison.

Shuffling states changes a different aspect of the system: the state vector is still supplied, but its correspondence to the trajectory is broken. The resulting macros, 73.3% and 50.2%, are close to or below greedy-value control and substantially below threshold-only control. This identifies the state–trajectory pairing as information that the policy uses productively. It also explains why the representation must be assessed together with the decision rule that consumes it. The fixed adaptive routing baseline in Table 1 is an external method comparison; threshold-only here is a policy replacement inside the shared CoT^2-Meta interfaces.

C.2 Counterfactual decision quality

The audit evaluates 900 non-terminal states. Each feasible action receives three stochastic continuations and the same remaining token allocation. Agreement measures alignment with the empirically best action among those continuations; normalized regret measures the associated downstream loss. Harmful-stop, harmful-prune, wasted-expansion, and repair-precision rates characterize specific control decisions.

Table 14: Counterfactual action audit on 900 states.

Policy	Agree.	Regret	Stop	Prune	Expand	Repair
Random feasible	29.8	0.342	13.8	16.4	35.2	44.8
Fixed-stage schedule	41.7	0.287	10.9	13.1	29.0	52.0
Greedy-value	49.6	0.253	8.7	10.8	24.1	60.4
UCB-only	53.2	0.228	7.8	9.4	21.8	63.2
Threshold-only	57.8	0.211	6.9	8.5	19.6	68.9
Shuffled meta-state	44.3	0.279	10.2	12.7	27.5	54.1
Full CoT^2-Meta	69.1	0.164	4.2	5.1	13.8	80.6

Agreement and repair precision are higher-is-better. Regret, harmful stopping, harmful pruning, and wasted expansion are lower-is-better. All rates except regret are percentages. Each feasible action receives the same remaining token allocation and three stochastic continuations.

Table 14 connects improved agreement to lower-cost mistakes. Against threshold-only control, agreement rises by 11.3 points and regret falls from 0.211 to 0.164, a 22.3% reduction. Harmful stopping falls by 39.1%, harmful pruning by 40.0%, and wasted expansion by 29.6%, relative to their respective threshold-only rates. These changes address both sides of allocation: prematurely terminating useful reasoning loses an available opportunity, whereas expanding an unproductive trajectory consumes tokens that could support another action. Repair precision rises by 11.7 points, indicating a more selective use of correction in the audit population.

The shuffled-state condition provides a complementary comparison. Its regret is 0.279 and repair precision 54.1%, versus 0.164 and 80.6% for the full policy. A mismatched state degrades the choice of what to preserve, where to spend further effort, and when to intervene. The audit metrics evaluate individual decision consequences, while Table 13 measures the final performance of complete policy runs. Their agreement supports the interpretation that better state-conditioned choices contribute to the end-to-end gains.

The best-action frequencies in Table 15 explain why a depth schedule is useful but coarse. Expansion is the early-depth mode at 54%, yet 46% of early states favor another action. In the middle group, pruning and expansion are nearly tied at 32% and 30%. At terminal depth, stopping accounts for 32% and repair for 27%; expansion and pruning still account for 32% together. Thus, the audit contains several computational needs at the same depth. Depth can summarize the stage of a search, while process quality, defect location, and confidence determine which action suits a partic-

Table 15: Offline best-action distribution across search-depth groups.

Depth	Expand	Prune	Repair	Stop	Abstain
Early	54	18	9	8	11
Middle	30	32	18	14	6
Terminal-depth	11	21	27	32	9

Each row sums to 100%. Rows report counterfactual best-action frequencies; Figure 5 reports the online controller's empirical action distribution.

ular trajectory. These frequencies describe offline best-action labels; Figure 5 displays the online controller's realized action distribution.

C.3 Component removals and signal fusion

Table 16: Component ablations at matched total-token expenditure on the three reasoning tasks.

Variant	MATH	GPQA	BBEH	Macro	Tokens	Calls
Full CoT^2-Meta	92.8	90.4	75.8	86.3	12,827	13.8
Without tree-structured interaction	86.0	82.1	69.7	79.3	12,810	12.9
Without process scoring	88.7	85.1	72.0	81.9	12,850	14.6
Without pruning	89.6	86.0	72.8	82.8	12,895	15.8
Without selective repair	90.1	86.8	73.2	83.4	12,820	14.1
Without adaptive stopping	91.3	88.1	74.1	84.5	12,800	15.9
Without abstention	92.2	89.7	75.2	85.7	12,860	13.8
Process-only control	90.0	87.0	73.8	83.6	12,900	14.2
Outcome-only control	89.2	85.9	72.9	82.7	12,790	14.0

Unused component-specific compute is reassigned to generation. The three-task macro and its token mean exclude LiveCodeBench.

Table 16 studies the functional parts of the loop at matched total-token expenditure, reassigning unused component-specific compute to generation. Removing tree interaction causes the largest loss, 7.0 macro points. Removing process scoring costs 4.4 points, pruning 3.5, and selective repair 2.9. These are individual removal contrasts against the full system: the components interact through the same search history. The tree creates alternative partial solutions; scoring distinguishes their quality; pruning and repair act on that distinction. Extra generation after a removal uses a similar allocation but produces a different distribution of useful computations.

The fused controller also exceeds both process-only control at 83.6% and outcome-only control at 82.7%. On GPQA, fusion gives 90.4%, versus 87.0% and 85.9%, respectively. Intermediate quality and terminal confidence therefore supply complementary inputs for the evaluated decisions. Removing adaptive stopping raises calls from 13.8 to 15.9 while lowering macro accuracy to 84.5%; completing more calls is an inefficient use of the matched token allocation in that variant. The abstention-removal row reaches 85.7% using the full example denominator. It evaluates a rerun control policy, whereas the nested-prefix analysis in Table 38 changes only which already-produced predictions are retained.

D Evaluator Validation and Signal Interventions

D.1 Held-out step labels and cross-domain agreement

Table 17: Step-level evaluator validation and earliest-error diagnostics.

Corpus / evaluator	AUROC	AUPRC	F1	Earliest error	Repair prec.
PRM800K held-out	0.881	0.812	0.766	0.739	0.801
Cross-domain manual set	0.846	0.784	0.742	0.711	0.785
Self-evaluator	0.846	0.784	0.742	0.711	0.785
GPT-4o judge	0.858	0.798	0.755	0.728	0.797
Qwen2.5-72B judge	0.823	0.751	0.714	0.688	0.752
Semantic-only	0.742	0.667	0.641	0.592	0.676
Logical-only	0.807	0.735	0.701	0.654	0.732
Random evaluator	0.501	0.425	0.431	0.207	0.447
Offline gold evaluator	0.952	0.927	0.891	0.864	0.914

PRM800K contributes 3,200 held-out steps. The cross-domain set contributes 1,200 manually labeled steps. The remaining online rows use that cross-domain set; the final row is the separate offline diagnostic.

Table 17 evaluates the signal at both a classification interface and an intervention interface. From the 3,200-step PRM800K set to the 1,200-step cross-domain set, the self-evaluator's AUROC changes from 0.881 to 0.846, F1 from 0.766 to 0.742, and earliest-error score from 0.739 to 0.711. The cross-domain scores retain substantial discrimination while revealing that locating the first defect is

a separate demand from ranking step quality. That distinction matters because a reasonably ranked trajectory can still receive an incorrectly positioned repair.

The signal-source contrasts reinforce this connection. Relative to logical-only scoring, the full self-evaluator improves the earliest-error score by 0.057 and repair precision by 0.053. GPT-4o changes them more modestly, from 0.711 to 0.728 and from 0.785 to 0.797. The offline gold-informed diagnostic reaches 0.864 and 0.914, respectively, indicating additional scope for improving the information supplied to the repair interface. These step-label diagnostics and the system-level replacement results in Table 3 evaluate complementary consequences of the same evaluator choices.

Table 18: Cross-domain human validation of the self-evaluator.

Domain	Steps	Error prev.	AUROC	AUPRC	F1	α
MATH	300	36.0%	0.872	0.808	0.763	0.86
GPQA	300	33.7%	0.844	0.786	0.746	0.82
BBEH	300	38.3%	0.826	0.754	0.720	0.79
LiveCodeBench	300	34.7%	0.842	0.788	0.737	0.83

Two annotators label every step independently and a third adjudicates disagreements. α denotes Krippendorff's agreement coefficient.

Table 18 allocates 300 manually labeled steps to each domain, holding validation-set size constant across the domain comparisons. BBEH has the highest error prevalence, 38.3%, the lowest evaluator AUROC, 0.826, and the lowest annotation agreement, $\alpha = 0.79$. MATH has AUROC 0.872 and $\alpha = 0.86$. This co-occurrence places BBEH at a more difficult point for both the labeled assessment and the automated signal; it motivates attention to the reliability of intermediate judgments on that task. GPQA and coding have similar AUROCs, 0.844 and 0.842, even though their candidate-availability profiles differ substantially in Table 25. Step discrimination alone therefore does not determine the final failure mix: the controller also operates on different candidate populations.

Two annotators label each step independently, with a third adjudicating disagreements. The reported α values characterize agreement before those labels are used to evaluate the process signal.

D.2 Evaluator source and controlled degradation

The replacement study in Table 3 fixes the generator and controller while changing the evaluator. The GPT-4o and Qwen2.5-72B outcomes remain close to the self-evaluator, whereas the semantic-only, logical-only, and randomized conditions alter the information available for action selection more substantially. The offline label-informed row defines a diagnostic evaluator condition.

Table 19: Controlled perturbation of process-evaluation scores.

Condition	AUROC	MATH	GPQA	BBEH	Macro
Full evaluator	0.846	92.8	90.4	75.8	86.3
Gaussian noise, $\sigma = 0.10$	0.829	92.4	89.9	75.4	85.9
Gaussian noise, $\sigma = 0.20$	0.795	91.6	88.8	74.5	85.0
Gaussian noise, $\sigma = 0.30$	0.747	90.2	87.0	73.0	83.4
Gaussian noise, $\sigma = 0.50$	0.653	87.7	83.9	70.4	80.7
Score permutation	0.506	84.0	79.8	68.1	77.3
Constant evaluator	0.500	84.8	80.7	68.8	78.1
Random evaluator	0.501	83.4	79.0	67.8	76.7

The generator, controller, prompts, candidates, and total-token allocation are held fixed. Noise magnitudes index the Gaussian-noise series; permutation, constant, and random signals are separate controls.

Table 19 reveals a nonuniform response to signal corruption. At $\sigma = 0.10$, the macro loss is 0.4 points; at $\sigma = 0.20$, it is 1.3; at $\sigma = 0.30$, it is 2.9; and at $\sigma = 0.50$, it is 5.6. The equal increase from $\sigma = 0.10$ to 0.20 costs 0.9 points, while the next equal increase costs 1.6. The controller tolerates mild perturbation better than the larger changes in this measured range. Across the endpoints, GPQA loses 6.5 points, compared with 5.1 on MATH and 5.4 on BBEH, showing task-dependent sensitivity to the quality of its control evidence.

The uninformative controls distinguish ranking information from the values that trigger actions. Permutation, constant, and random scores have AUROC near 0.5, yet their final macros are 77.3%, 78.1%, and 76.7%. The constant condition exceeds both varying uninformative signals. This pattern is consistent with harmful action changes induced by misleading score variation: the controller uses score levels as well as their ordering, through pruning, repair, and stopping conditions. The experiment supports the value of trajectory-aligned information and shows that AUROC alone is an incomplete summary of a score's usefulness inside the control loop.

E Token Scaling, Candidate Coverage, and Failure Attribution

E.1 Complete task-wise scaling results

The four scaling matrices compare how methods use increasing allocations relative to the same task–backbone reference. All use Claude-4.5-Opus, and their $8\times$ rows supply the corresponding task results in Table 1. Reading the intermediate allocations as well as the endpoints distinguishes sustained selection gains from additional search opportunity.

Table 20: MATH accuracy as a function of total-token budget.

Budget	BoN	Verifier	PRM-BoN	ToT	ReST	Fixed	CoT^2-Meta
$2\times$	80.5	81.4	82.5	82.0	82.8	83.1	84.4
$4\times$	82.8	84.0	86.0	84.8	85.9	86.4	88.2
$8\times$	84.2	86.8	89.4	87.4	89.2	90.0	92.8
$16\times$	85.8	88.0	91.0	89.3	90.8	91.5	94.1

Budget is expressed in multiples of B_{greedy} for this task and backbone. All rows use Claude-4.5-Opus.

Table 20 separates the value of processing a trajectory from simply obtaining additional samples. At $8\times$, verification adds 2.6 points to Best-of-N, PRM-guided selection adds another 2.6, and the full controller exceeds PRM-guided selection by 3.4. These are contrasts between complete inference methods, with the policy-isolation study providing the shared-component comparison. The full controller reaches 92.8% at $8\times$, above the strongest $16\times$ baseline at 91.5%. Its own gain from $4\times$ to $8\times$ is 4.6 points, versus 1.3 from $8\times$ to $16\times$. At the larger allocation, additional computation converts fewer remaining MATH problems into correct answers. The high candidate availability in Table 25 makes selection and local preservation important alongside further sampling.

Table 21: GPQA accuracy as a function of total-token budget.

Budget	BoN	Verifier	PRM-BoN	ToT	ReST	Fixed	CoT^2-Meta
$2\times$	76.8	77.5	79.0	78.2	79.4	79.8	81.3
$4\times$	78.3	79.8	82.2	81.0	82.8	83.6	86.0
$8\times$	79.6	82.2	85.6	83.5	85.2	86.1	90.4
$16\times$	81.5	84.0	87.1	85.2	86.8	88.0	92.0

Budget is expressed in multiples of B_{greedy} for this task and backbone. All rows use Claude-4.5-Opus.

The GPQA matrix in Table 21 shows the largest widening of the margin over fixed routing up to $8\times$: 1.5 points at $2\times$, 2.4 at $4\times$, and 4.3 at $8\times$. The margin remains 4.0 points at $16\times$. Additional budget benefits both methods, but its allocation increasingly distinguishes their outputs through the middle of this range. At $8\times$, the full controller also exceeds PRM-guided sampling by 4.8 points. In the designated audit, a correct candidate exists for 194 of 198 GPQA problems, while only 179 outputs are correct. This combination links the scaling result to an identifiable selection problem: many residual errors occur in searches that already contain a correct candidate. Larger candidate pools and better management of their competing evidence are different demands on the same allocation.

Table 22 illustrates why the comparison must follow the strongest task-specific method at each allocation. Fixed routing leads the baselines at $2\times$ and $4\times$, whereas ReST-MCTS* is strongest at $8\times$ and $16\times$. The full controller exceeds these respective baselines by 0.9, 1.7, 2.0, and 2.3 points. Its incremental gains per doubling decline from 3.4 to 2.6 to 1.3 points, with accuracy

Table 22: BBEH accuracy as a function of total-token budget.

Budget	BoN	Verifier	PRM-BoN	ToT	ReST	Fixed	CoT^2-Meta
2×	66.8	67.3	68.0	68.0	68.6	68.9	69.8
4×	68.0	69.0	71.2	69.9	71.0	71.5	73.2
8×	68.9	70.5	73.5	71.2	73.8	73.4	75.8
16×	70.0	72.0	74.6	72.8	74.8	74.7	77.1

Budget is expressed in multiples of B_{greedy} for this task and backbone. All rows use Claude-4.5-Opus.

reaching 77.1% at the largest allocation. The persistent residual population aligns with two findings elsewhere: BBEH has lower evaluator discrimination in Table 18, and 145 of its 242 designated-run errors lack a correct candidate. Both the available proposal set and the reliability of its intermediate assessment remain relevant to BBEH control.

Table 23: LCB accuracy as a function of total-token budget.

Budget	BoN	Verifier	PRM-BoN	ToT	ReST	Fixed	CoT^2-Meta
2×	44.2	45.0	46.2	45.4	46.5	46.8	48.1
4×	46.0	47.5	49.5	48.3	50.4	50.8	52.2
8×	47.8	49.8	53.4	51.2	53.2	53.9	56.7
16×	49.5	52.0	55.8	53.0	55.1	55.6	58.2

Budget is expressed in multiples of B_{greedy} for this task and backbone. All rows use Claude-4.5-Opus.

In Table 23, the full controller reaches 56.7% at $8\times$, exceeding the strongest baseline at $16\times$, PRM-guided sampling at 55.8%. Doubling its own allocation to $16\times$ adds only 1.5 points, despite a substantial residual error rate. The proposal audit gives a concrete interpretation of this slow return: only 616 of 880 problems contain a correct candidate, and 117 of those opportunities are lost before final output. Substantial unresolved error thus coexists with small gains from additional search, emphasizing the quality of new executable solutions and their selection. The hard-slice study in Table 40 provides a different measured allocation-response profile.

E.2 Candidate availability and final selection

Candidate coverage measures the existence of at least one correct generated candidate. Final accuracy measures the correctness of the returned answer. Their difference, the selection gap, counts available opportunities that are lost before output. The three-task macro comparison and the designated-run counts below assess these quantities at their respective aggregation levels.

Table 24: Candidate coverage and selection quality across MATH, GPQA, and BBEH.

Method	Coverage	Accuracy	Gap	Branches	Duplicate
Best-of-N	87.8	77.6	10.2	10.2	33.5
Vanilla ToT	89.9	80.7	9.2	8.7	28.0
ReST-MCTS*	91.5	82.7	8.8	9.4	24.5
CoT^2-Meta without strategy switching	91.7	83.6	8.1	8.6	25.2
Full CoT^2-Meta	93.5	86.3	7.2	12.8	11.2

Coverage, accuracy, and duplicate rate are percentages; gap is coverage minus accuracy, in percentage points. Branches denotes the reported effective-branch count. Similarity is a post-hoc diagnostic of the generated trajectories.

Table 24 gives an accounting of the full controller's 3.6-point advantage over ReST-MCTS*:

$$\underbrace{86.3 - 82.7}_{\text{accuracy difference}} = \underbrace{93.5 - 91.5}_{\text{coverage difference}} + \underbrace{8.8 - 7.2}_{\text{selection-gap reduction}} .$$

The observed difference contains both broader access to correct candidates and better retention of that access through final selection. The quantities arise from the complete search policies and describe jointly determined outcomes.

Removing strategy switching yields 91.7% coverage and 83.6% accuracy. Restoring it adds 1.8 coverage points and reduces the selection gap by 0.9 points, while the effective-branch count changes from 8.6 to 12.8 and duplication falls from 25.2% to 11.2%. The 48.8% increase in effective branches accompanies a more varied set of continuations and more correct-candidate opportunities. These are properties of the generated pool, measured by post-hoc similarity. The active method manages branches through generation, scoring, pruning, and repair; its semantic-similarity statistic is a diagnostic of the resulting search.

Table 25: Candidate and output counts in the designated audit run.

Dataset	N	Output	Candidate	Cov.	Acc.	Gap
MATH-500	500	464	485	97.0	92.8	4.2
GPQA-Diamond	198	179	194	98.0	90.4	7.6
BBEH fixed slice	1000	758	855	85.5	75.8	9.7
LiveCodeBench release_v5	880	499	616	70.0	56.7	13.3

Output counts examples answered correctly. Candidate counts examples with at least one correct generated candidate. These counts come from the designated audit run; Appendix H reports five-seed aggregation.

Table 25 shows that the same aggregate gap conceals different task-level losses. On MATH, 21 of 485 searches with a correct candidate fail to return one; on GPQA, the count is 15 of 194. These account for 58.3% and 78.9% of the respective residual errors. On BBEH and LiveCodeBench, absent candidates instead account for 145/242 and 264/381 errors, or 59.9% and 69.3%. The conditional conversion of an available candidate into a correct output is 95.7% on MATH, 92.3% on GPQA, 88.7% on BBEH, and 81.0% on LiveCodeBench, computed from the designated-run counts.

These contrasts distinguish useful computational objectives. GPQA places substantial pressure on choosing and preserving an already-generated correct answer. Coding has both a larger proposal deficit and the lowest conversion rate among candidate-containing searches. The controller's action set spans those needs: expansion changes candidate availability, while selection and repair determine how much of that availability reaches the output. The counts describe one designated audit run; the five-seed comparisons in Appendix H describe repeated complete evaluations.

Table 26: Three-task coverage and final-accuracy scaling.

Budget	ReST cov.	ReST acc.	CoT^2-Meta cov.	CoT^2-Meta acc.
2×	86.0	76.9	85.5	78.5
4×	89.3	79.9	90.5	82.5
8×	91.5	82.7	93.5	86.3
16×	93.3	84.1	94.8	87.7

Rates are macro-averaged over MATH, GPQA, and BBEH. Coverage is the existence of a correct candidate; final accuracy measures its eventual selection.

The low-budget rows of Table 26 isolate the importance of conversion particularly clearly. At $2\times$, the full controller has 0.5 points less coverage than ReST-MCTS* but 1.6 points more accuracy; its selection gap is 7.0 rather than 9.1 points. At $4\times$, its advantage comprises 1.2 additional coverage points and a 1.4-point gap reduction. By $8\times$, both contributions have grown to 2.0 and 1.6 points. Better use of an available pool appears before the full policy obtains the larger pool.

From $8\times$ to $16\times$, the full controller's coverage grows by 1.3 points and accuracy by 1.4, leaving a nearly unchanged gap of 7.1 points. ReST-MCTS* gains 1.8 coverage points but only 1.4 accuracy points, so its gap widens to 9.2. At this end of the measured range, additional candidate availability has a limited conversion into final success. The result complements the task-wise diminishing returns: further allocation should be judged by both the solutions it discovers and the opportunities it preserves.

E.3 Residual-error decomposition

The counts give two accounting identities for the designated audit:

$$(500 - 464) + (198 - 179) + (1000 - 758) + (880 - 499) = 678,$$

$$(500 - 485) + (198 - 194) + (1000 - 855) + (880 - 616) = 428.$$

The first is the number of incorrect outputs, and the second is the number of searches without a correct candidate. Their difference is 250 examples with a correct candidate but an unsuccessful final outcome.

Table 27: Mutually exclusive attribution of the 678 residual errors in the designated audit run.

Failure source	Count	Share
Proposal miss	428	63.1%
Evaluator misranking	90	13.3%
Controller decision error	71	10.5%
Repair / continuation damage	45	6.6%
Budget exhaustion	44	6.5%
Total	678	100.0%

Proposal miss refers to absence of a correct candidate in the evaluated search. The other categories follow the audit attribution.

Table 27 locates losses at different stages of the reasoning loop. Proposal misses form 63.1% of all errors, while evaluator misranking and controller decision errors together contribute 161 cases. Among the 250 errors after candidate availability, those two categories account for 64.4%; repair or continuation damage accounts for 18.0%, and budget exhaustion for 17.6%. The audit therefore identifies a substantial population in which assessing and acting on existing reasoning is the relevant failure stage, even though proposal absence is the largest aggregate category.

The task composition matters for that aggregate. BBEH and LiveCodeBench contribute 623 of 678 errors and 409 of 428 proposal misses. The four-task macro in Table 1 weights tasks equally, while this count decomposition weights failed examples. Read together, the two views support a task-dependent allocation objective: proposal diversity has greater scope on BBEH and coding, whereas preservation and selection address most observed GPQA errors. A proposal miss denotes the absence of a correct candidate in the evaluated search history; the decomposition characterizes that history and its downstream decisions.

F Local-Repair Mechanism

Table 28: Token-matched repair strategies on MATH-500.

Strategy	Acc.	Tokens	Trigger	Rescue	Hurt	Prefix	Tok./rescue
No repair + extra expansion	90.1	15,452	0.0	–	–	–	–
Full-trajectory regeneration	89.4	15,510	31.8	28.6	18.9	0.0	1130
Fixed one-time repair	90.5	15,395	20.0	34.2	13.8	76.5	890
Repair-all	90.8	15,488	100.0	38.1	17.2	84.0	980
Selective local repair	92.8	15,440	32.4	52.4	6.8	94.2	610

Trigger, rescue, hurt, and prefix preservation are trajectory-level percentages. System accuracy is example-level. Rescue and hurt use their corresponding repair-triggered trajectory populations.

Table 28 distinguishes the frequency, extent, and location of correction. Selective local repair and full-trajectory regeneration are triggered at similar rates, 32.4% and 31.8%, yet their system accuracies differ by 3.4 points. Selective repair preserves 94.2% of prefixes, with a 52.4% rescue rate and 6.8% hurt rate; regeneration preserves none, with corresponding rates of 28.6% and 18.9%. This combination supports retaining useful partial reasoning while revising the identified suffix. Triggering a revision with roughly similar frequency can have very different consequences when the amount of preserved reasoning changes.

Repairing every trajectory produces 90.8% accuracy, 2.0 points below selective repair, with a 17.2% hurt rate. A fixed one-time repair reaches 90.5%, and reallocating repair compute to expansion

reaches 90.1%. The selective policy therefore outperforms both indiscriminate intervention and spending the allocation on additional proposals. Its reported cost per rescue is 610 tokens, 46.0% lower than regeneration's 1,130. Rescue, hurt, trigger, and prefix-preservation rates describe their corresponding trajectory populations; the system-accuracy column evaluates final answers over MATH-500. Together they connect the quality of an intervention to its usefulness inside a multi-trajectory search.

Table 29: Repair outcomes grouped by earliest-error localization distance.

$\lvert\hat{i} - i^\star\rvert$	Audited triggers	Success	Harm
0	180	68%	3%
1	118	54%	7%
2	72	35%	14%
≥ 3	50	16%	26%

The audit contains 420 triggers. Success and harm rates are rounded to whole percentages.

Table 29 identifies localization as a consequential part of the repair interface. Of 420 audited triggers, 298, or 71.0%, are located within one step of the reference error. Exact localization is associated with 68% repair success and 3% harm. At distance one, success is 54%; at distance two, 35%; and at distance at least three, 16%, with harm reaching 26% in that last group. The far-localization group contains 50 triggers, or 11.9% of the audit, yet exhibits a sharply different repair profile from the exactly localized group.

The mechanism implied by prefix-based revision explains why location matters. Revising too early discards reasoning before the defect; revising too late preserves a faulty step in the prefix. The distance groups pool these two directions and associate increasing localization error with poorer outcomes. This supports locating the intervention point as well as deciding whether to repair. It also gives a concrete interpretation of the evaluator's earliest-error diagnostic in Table 17. Success and harm percentages are reported at whole-percentage precision.

G Development Selection, Transfer, and Sensitivity

The development procedure evaluates 88 configurations: a 64-point Sobol design followed by 24 local refinements. Baselines receive the same development-token allowance. The selected settings are frozen before test evaluation.

Table 30: Reported development ranges and selected shared defaults.

Parameter	Minimum	Maximum	Default	Selection criterion
Fusion λ	0.20	0.60	0.40	Macro utility
Exploration β	0.75	1.75	1.25	Macro utility
Pruning τ_{prune}	0.25	0.45	0.35	Macro utility
Repair θ_{health}	0.40	0.60	0.50	Macro utility
Stopping τ_{stop}	0.85	0.95	0.90	Risk–cost utility
Abstention τ_{abs}	0.60	0.80	0.70	Selective risk

Table 30 separates the objectives of allocating and terminating reasoning. Fusion, exploration, pruning, and repair are selected by development macro utility; stopping uses risk–cost utility; abstention uses selective risk. These criteria reflect distinct consequences of an action: expansion and repair seek an improved answer, while stopping and abstention govern whether an existing answer should be emitted. Every reported shared default lies inside its development range. The choice is therefore assessed together with the transfer and local-response measurements, which show how this operating point behaves when the deployment task or parameter setting changes.

Table 31 separates specialization to a task from using a shared configuration. Cross-domain-development freezing changes the four task scores by only 0.1–0.3 points relative to per-task tuning. Math-development freezing costs 0.2 points on MATH but 0.6 on GPQA and BBEH and 0.7 on LiveCodeBench, a pattern consistent with the benefit of representing multiple task regimes during

Table 31: Transfer of development-selected controller thresholds.

Configuration	MATH	GPQA	BBEH	LCB	Macro
Per-task tuned	92.8	90.4	75.8	56.7	78.9
Math-dev frozen	92.6	89.8	75.2	56.0	78.4
Cross-domain-dev frozen	92.7	90.2	75.6	56.4	78.7
Global cross-model frozen	92.1	89.7	75.1	55.9	78.2

All threshold selection uses development data. Test-set evaluation follows configuration freezing.

development. Global cross-model freezing incurs losses of 0.7, 0.7, 0.7, and 0.8 points. Even this configuration remains above the strongest task-specific external baselines in Table 1 on each task.

The 0.7-point displayed macro penalty under global freezing is much smaller than the 5.6-point Claude loss under shuffled states. The two interventions probe different dependencies: threshold freezing tests transport of the operating point, whereas shuffling disrupts the information supplied at each decision. Their relative effects support the importance of preserving the trajectory–action relationship while using a broadly shared threshold configuration.

Table 32: One-at-a-time threshold sensitivity on the three reasoning tasks.

Parameter	Grid 1	Grid 2	Default	Grid 4	Grid 5
Fusion weight	0.32 / 85.7	0.36 / 86.1	**0.4 / 86.3**	0.44 / 86.1	0.48 / 85.8
Exploration weight	1 / 85.8	1.125 / 86.1	**1.25 / 86.3**	1.375 / 86.2	1.5 / 85.9
Pruning threshold	0.28 / 85.6	0.315 / 86.1	**0.35 / 86.3**	0.385 / 86.1	0.42 / 85.7
Repair health	0.4 / 85.6	0.45 / 86.0	**0.5 / 86.3**	0.55 / 86.2	0.6 / 85.8
Stopping threshold	0.85 / 85.9	0.875 / 86.2	**0.9 / 86.3**	0.925 / 86.1	0.95 / 85.7
Abstention threshold	0.6 / 86.1	0.65 / 86.2	**0.7 / 86.3**	0.75 / 86.2	0.8 / 86.1

Each entry is parameter value / macro accuracy. The center column gives the reported shared defaults.

The local sweeps in Table 32 show different parameter roles. Abstention changes macro accuracy by at most 0.2 points over the reported grid, whereas pruning and repair health each vary by 0.7. Lowering the repair-health threshold from 0.50 to 0.40 costs 0.7 points; raising it to 0.60 costs 0.5. These are the two sides of deciding which defects receive intervention. Stopping at 0.85 gives 85.9%, and at 0.95 gives 85.7%, versus 86.3% at 0.90, identifying an interior operating point in the measured range. Earlier emission and prolonged search have different costs for the same trajectory-management objective.

Fusion and exploration also have broad local neighborhoods: each adjacent fusion value gives 86.1%, and the adjacent exploration values give 86.1% and 86.2%. Moderate adjustments preserve most of the measured score, while component removal in Table 16 produces larger changes. The sweep measures local parameter response with the rest of the controller fixed; the joint matrix below examines changes to two coupled decisions.

Table 33: Joint sensitivity of pruning and process/outcome fusion.

$\tau_{\text{prune}} \backslash \lambda$	0.32	0.36	0.40	0.44	0.48
0.28	84.9	85.3	85.6	85.4	85.0
0.315	85.4	85.9	86.1	85.9	85.5
0.35	85.7	86.1	86.3	86.1	85.8
0.385	85.5	85.9	86.1	85.8	85.4
0.42	85.0	85.4	85.7	85.4	84.9

Entries are three-task macro accuracy. The center row and column reproduce the corresponding one-at-a-time sweeps.

Table 33 couples the construction of the branch value to the threshold applied to that value. Its central 3×3 region, spanning $\lambda \in [0.36, 0.44]$ and $\tau_{\text{prune}} \in [0.315, 0.385]$, ranges from 85.8% to 86.3%. The default thus lies inside a neighborhood with a maximum 0.5-point loss. The outer corners fall to 84.9–85.0%, with losses of 1.3–1.4 points. For example, changing fusion alone to 0.48 gives 85.8%

and pruning alone to 0.42 gives 85.7%, whereas changing both gives 84.9%. The observed joint loss exceeds either individual loss. Fusion controls the evidence assigned to a branch and pruning controls its retention; interpreting their effects together follows directly from this shared decision pathway. The center row and column also retain the corresponding one-at-a-time comparisons.

H Five-Seed Outcomes and Paired Comparisons

For a task with N examples and correct-answer counts $c_1, \ldots, c_5$, the pooled percentage is $100 \sum_{s=1}^{5} c_s/(5N)$. Each seed uses the same task population. The comparator is fixed adaptive routing, except on Claude/BBEH, where ReST-MCTS* is strongest.

Table 34: Integer seed outcomes: Claude.

Task	Method	N	S1	S2	S3	S4	S5	Pooled	Acc.
MATH	Baseline	500	448	450	449	452	451	2250/2500	90.0
MATH	CoT^2-Meta	500	462	464	465	463	466	2320/2500	92.8
GPQA	Baseline	198	170	171	169	170	172	852/990	86.1
GPQA	CoT^2-Meta	198	178	179	180	179	179	895/990	90.4
BBEH	Baseline	1000	734	739	740	736	741	3690/5000	73.8
BBEH	CoT^2-Meta	1000	754	759	761	756	760	3790/5000	75.8
LCB	Baseline	880	472	474	475	475	476	2372/4400	53.9
LCB	CoT^2-Meta	880	497	499	500	498	501	2495/4400	56.7

Seed cells are correct-answer counts. Each task uses the same N in every seed. Baseline is fixed adaptive routing, except Claude/BBEH, which uses ReST-MCTS*.

Table 34 shows positive count differences at every listed seed on all four tasks. MATH gains range from 11 to 16 correct answers per seed, GPQA from 7 to 11, BBEH from 19 to 21, and LiveCodeBench from 23 to 25. Across the five runs, the net count differences are 70, 43, 100, and 123. Relative to the corresponding baseline error totals, these remove 28.0%, 31.2%, 7.6%, and 6.1% of errors. The same controller therefore has a comparatively large effect on the remaining mathematics and scientific-QA errors, while coding contributes more additional correct outputs in absolute count because its evaluation population is larger.

Table 35: Integer seed outcomes: Llama.

Task	Method	N	S1	S2	S3	S4	S5	Pooled	Acc.
MATH	Baseline	500	312	315	316	314	316	1573/2500	62.9
MATH	CoT^2-Meta	500	329	331	333	330	332	1655/2500	66.2
GPQA	Baseline	198	108	109	110	108	109	544/990	54.9
GPQA	CoT^2-Meta	198	114	115	116	114	115	574/990	58.0
BBEH	Baseline	1000	609	613	617	611	615	3065/5000	61.3
BBEH	CoT^2-Meta	1000	637	641	645	639	643	3205/5000	64.1
LCB	Baseline	880	280	282	283	282	285	1412/4400	32.1
LCB	CoT^2-Meta	880	309	310	311	311	312	1553/4400	35.3

Seed cells are correct-answer counts. Each task uses the same N in every seed. Baseline is fixed adaptive routing, except Claude/BBEH, which uses ReST-MCTS*.

The Llama counts in Table 35 show the same positive direction in every listed seed. MATH gains are 16–17 correct answers per run, GPQA gains six, BBEH gains 28, and LiveCodeBench gains 27–29. The pooled gains are $82/2500$, $30/990$, $140/5000$, and $141/4400$, expressed as fractions of the repeated task evaluations. The GPQA difference is computed before rounding: $100(574 - 544)/990 = 3.0303\ldots$ points, reported as 3.0. These counts connect the task scores to the repeated-run evidence and keep the small-sample scientific-QA comparison on its actual integer outcome scale.

Table 36 reports the paired analysis of evaluation instances, using 10,000 bootstrap resamples and Holm adjustment over four tasks within each backbone. All eight intervals lie above zero, and the largest adjusted p-value is 0.004. The lower confidence bounds range from 1.1 to 2.1 points, so the

Table 36: Paired comparison with the strongest task-specific external baseline.

Backbone	Task	Baseline	CoT^2-Meta	Δ (pp)	95% CI	Adj. p
Claude	MATH	90.0	92.8	+2.8	[1.6, 4.0]	< 0.001
Claude	GPQA	86.1	90.4	+4.3	[2.0, 6.7]	0.001
Claude	BBEH	73.8	75.8	+2.0	[1.1, 2.9]	0.002
Claude	LCB	53.9	56.7	+2.8	[1.7, 3.9]	< 0.001
Llama	MATH	62.9	66.2	+3.3	[2.1, 4.5]	< 0.001
Llama	GPQA	54.9	58.0	+3.0	[1.1, 5.1]	0.004
Llama	BBEH	61.3	64.1	+2.8	[1.7, 3.9]	< 0.001
Llama	LCB	32.1	35.3	+3.2	[2.0, 4.4]	< 0.001

Point estimates are computed from the integer five-seed totals before rounding. Intervals and Holm-adjusted p-values are paired-bootstrap estimates (10,000 instance resamples; four tests within each backbone).

evidence supports a positive advantage across both backbones and every core task under the reported procedure. GPQA has wider intervals, 4.7 points with Claude and 4.0 with Llama, than the 1.8- and 2.2-point BBEH intervals. This quantifies the greater uncertainty in the magnitude of the GPQA advantage, alongside its smaller evaluation population.

The seed tables and paired intervals answer complementary questions. Seed counts expose the repeated direction and magnitude of the net correct-answer gain. Paired instance outcomes determine how the two methods' successes overlap and enter the uncertainty calculation. The reported intervals use that pairing; the count tables supply the marginal totals and exact point estimates.

I Selective Prediction, Calibration, and Additional-Budget Evaluation

I.1 Calibration and frozen confidence rankings

Table 37: Confidence calibration on MATH-500 at $8B_{\text{greedy}}$.

Method	Accuracy ↑	ECE ↓	Brier ↓
Best-of-N	84.2	0.118	0.142
ReST-MCTS*	89.2	0.073	0.103
Fixed routing	90.0	0.061	0.094
CoT^2-Meta	92.8	0.030	0.072

All methods share the confidence-extraction interface and use a development-selected calibration map.

Table 37 evaluates the confidence accompanying the selected MATH answers, using a shared extraction interface and a development-selected calibration map. Relative to fixed routing, CoT^2-Meta reduces ECE by 50.8% and Brier score by 23.4%, while increasing accuracy by 2.8 points. The two confidence metrics measure different aspects of the predictions: ECE summarizes the agreement between confidence and empirical correctness, while Brier score also reflects the probabilities assigned to individual outcomes. Their joint improvement gives the emission decision a better-calibrated score in this evaluation. The direct value of using that score for withholding is measured by the nested prediction sets below.

Table 38 shows that the remaining errors are strongly concentrated in the low-confidence tail. The first 460 predictions contain 456 correct answers and four errors; the last 40 contain only eight correct answers and 32 errors. Withholding that 8% tail therefore removes 88.9% of all 36 errors. The final 20 predictions are even more concentrated, containing one correct answer and 19 errors. These increments, obtained by subtracting neighboring prefix counts, characterize where the confidence ranking places the unresolved cases.

At the top-442 operating point, withholding 58 examples removes 33 errors and 25 correct answers. Selective accuracy is $439/442 = 99.3\%$, coverage is $442/500 = 88.4\%$, and the correct-answer contribution over the full set is $439/500 = 87.8\%$. Thus, a high-precision emission policy trades

Table 38: Nested confidence-prefix counts on MATH-500.

Coverage	k	Best-of-N	ReST	Fixed	CoT^2-Meta
80.0	400	380/400 (95.0)	391/400 (97.8)	394/400 (98.5)	399/400 (99.8)
88.4	442	408/442 (92.3)	430/442 (97.3)	434/442 (98.2)	439/442 (99.3)
92.0	460	415/460 (90.2)	438/460 (95.2)	442/460 (96.1)	456/460 (99.1)
96.0	480	419/480 (87.3)	444/480 (92.5)	448/480 (93.3)	463/480 (96.5)
100.0	500	421/500 (84.2)	446/500 (89.2)	450/500 (90.0)	464/500 (92.8)

Each entry is correct/retained (selective accuracy, %). Every method uses one frozen confidence ordering throughout the curve.

5.4% of the otherwise correct answers for removal of 91.7% of the errors. At the same retained-set size, fixed routing has eight errors, ReST-MCTS* has 12, and Best-of-N has 34, compared with three for CoT^2-Meta. This matched-coverage comparison evaluates the combination of answer quality and the confidence ordering actually used to retain predictions.

I.2 Hybrid fallback with additional tokens

Table 39: Selective prediction and additional-budget fallback on MATH-500.

Configuration	Correct/k	Full acc.	Cov.	Sel. acc.	ECE	Fallback	Tokens
Standalone	464/500	92.8	100.0	92.8	0.030	0.0	15,440
Selective top-442	439/442	87.8	88.4	99.3	0.019	0.0	15,440
Hybrid low threshold	470/500	94.0	100.0	94.0	0.028	18.0	18,190
Hybrid high threshold	473/500	94.6	100.0	94.6	0.024	34.0	20,710

The selective row is a prefix of the standalone confidence ranking. Hybrid rows use rerouting with additional tokens and constitute separate operating policies.

Table 39 distinguishes withholding existing predictions from allocating additional reasoning. The selective prefix retains 439 correct answers at the same 15,440 mean tokens as standalone inference. The low-threshold cascade returns 470 correct answers, six more than standalone, at 18,190 tokens; the high-threshold cascade returns 473 at 20,710 tokens. Mean expenditure increases by 17.8% and 34.1%, respectively. These cascades use new inference, whereas the selective row changes the emission set of a fixed ranking.

The larger cascade has a lower marginal return over the smaller cascade. Standalone to low-threshold routing adds 1.2 accuracy points for 2,750 additional mean tokens, approximately 0.44 points per 1,000 added mean tokens. Low to high adds 0.6 points for another 2,520 tokens, approximately 0.24 points per 1,000. Fallback rates of 18% and 34% correspond to 90 and 170 routed examples in the 500-problem evaluation, while the net extra correct outputs are six and nine. These aggregate results make the decision to continue reasoning concrete: the confidence-based routing policy should be judged by the final answers improved per added allocation, as well as by its fallback frequency.

I.3 LiveCodeBench-Plus pilot

Table 40: CoT2-Meta scaling on the 91-problem LiveCodeBench-Plus slice.

Budget multiplier	Pass@1 (%)
2×	39.6
4×	50.5
8×	60.4
16×	67.0

The pilot reports Greedy Pass@1 of 27.5%, generator pass@16 of 69.2%, and 8× proposal coverage of 72.5%. The 8× selection gap is 12.1 percentage points.

Table 40 gives a different scaling profile on the 91-problem LiveCodeBench-Plus slice. Successive allocation doublings add 10.9, 9.9, and 6.6 Pass@1 points, reaching 67.0% at $16\times$. The upper-range gain is larger than the 1.5-point gain in the 880-problem LiveCodeBench evaluation. The two populations consequently offer different returns to further search; a single budget multiplier should be read together with the task's measured response to additional computation.

The pilot's $8\times$ proposal coverage of 72.5% and final Pass@1 of 60.4% leave a 12.1-point selection gap. About 83.3% of candidate-containing opportunities are converted into correct outputs on the reported percentage scale, with the rest lost before final selection. The reported generator pass@16 is 69.2% and Greedy Pass@1 is 27.5%, providing additional views of the proposal regime. The coverage and scaling results jointly identify room for both discovering and selecting successful programs. This pilot slice complements the complete release_v5 results.

J Additional Visual Analyses

J.1 Cross-model macro-metric profiles

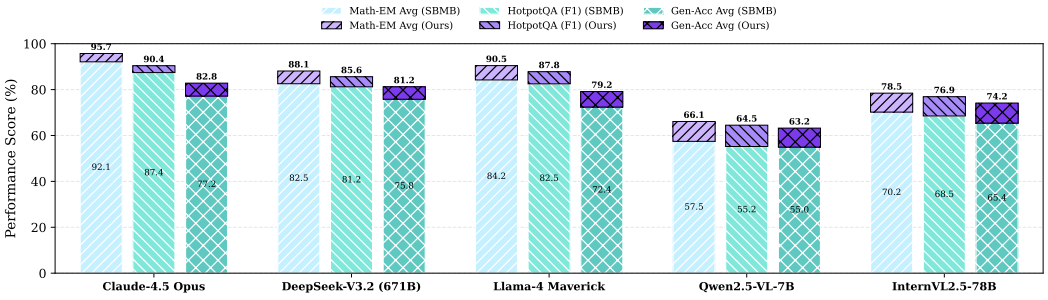


Figure 6: **Cross-model macro-metric summary under a fixed inference budget ($C = 16$).** For each backbone, we report three aggregated dimensions: Math-EM (average of GSM8K and MATH), QA-F1 (HotpotQA F1), and Gen-Acc (macro-average over GPQA, MMLU-Pro, BBEH, TruthfulQA, and LiveCodeBench). In each bar group, the base segment corresponds to the strongest budget-matched baseline score, and the top segment indicates the additional gain of CoT^2-Meta over that baseline.

Figure 6 evaluates within-backbone gains under its $C = 16$ call-budget protocol. The mathematical-score increments are 3.6 points for Claude, 5.6 for DeepSeek, 6.3 for Llama-4 Maverick, 8.6 for Qwen2.5-VL-7B, and 8.3 for InternVL2.5-78B. The HotpotQA increments are 3.0, 4.4, 5.3, 9.3, and 8.4, respectively. The lower-scoring Qwen and InternVL configurations exhibit larger absolute gains in these displayed comparisons, locating substantial opportunity for inference-time control even when the starting task score is lower. All three displayed metric groups improve within every backbone, so the result extends across different output objectives as well as model identities.

The general-task group adds 5.6 points for Claude and 8.2 for Qwen, while their mathematical gains are 3.6 and 8.6. The benefit varies across metric groups, with each backbone assessed on different output objectives. The figure documents those end-to-end profiles; Table 13 tests action selection under shared components on Claude-4.5-Opus and Llama-3.1-8B-Instruct, and Table 1 reports their total-token operating points. This places cross-model breadth alongside the direct policy intervention.

J.2 Budget, robustness, and meta-state profiles

Figure 7 compares allocation changes with perturbations of the reasoning interface. Within S1–S3, GPQA gains over the plotted strongest budget-matched baseline are 5.5, 7.3, and 6.9 points at 4, 8, and 16 calls; GSM8K gains are 2.1, 1.9, and 1.8. Additional calls therefore interact with the task's starting performance and available reasoning opportunities. The within-panel margins carry this comparison; each panel's radial scale and baseline correspond to the evaluation displayed there.

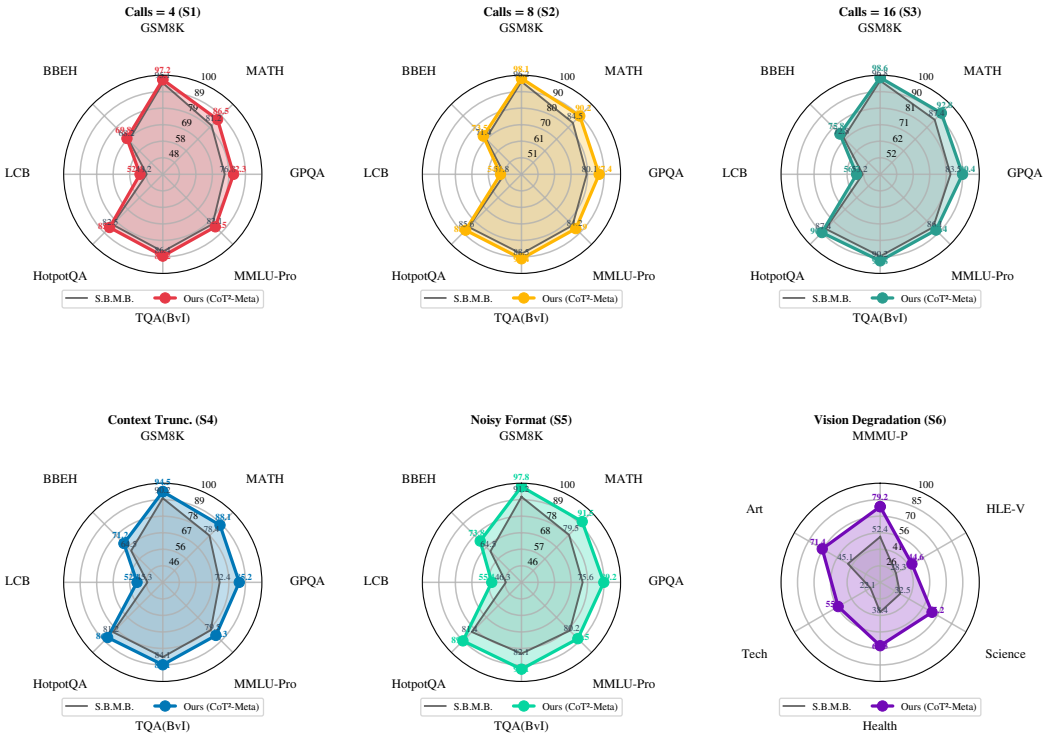


Figure 7: **Radar profiles of CoT^2-Meta across efficiency, robustness, and vision degradation.** S1–S3 evaluate budget scaling under Calls=$\{4, 8, 16\}$ on eight benchmarks. S4 applies context truncation to intermediate scratchpads (50% truncation). S5 injects formatting noise. S6 evaluates textified vision benchmarks under OCR and image degradation. In each subplot, S.B.M.B. denotes the strongest budget-matched baseline.

The perturbation panels reveal a different pressure on control. Under context truncation, the MATH and GPQA margins are 9.7 and 12.8 points; under formatting noise they are 12.0 and 13.6. These margins concern the end-to-end system when the supplied reasoning context or its presentation is disrupted. In the vision-degradation panel, the margins are 26.8 points on MMMU-P and 16.3 on HLE-V, evaluating downstream reasoning through the textification interface under degraded inputs. The panels show resilience of the combined inference interface in these conditions. The score-corruption experiment in Table 19 supplies the controlled evidence connecting process information to action quality, while the radar profiles show how complete-system performance behaves under broader perturbations.

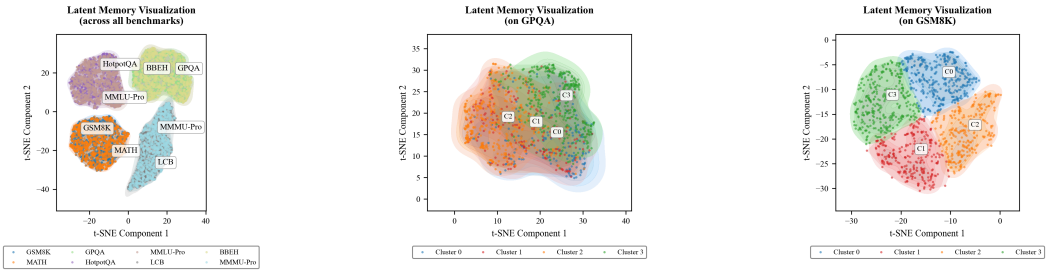


Figure 8: **Latent memory visualization.** Left: across all benchmarks (GSM8K, MATH, GPQA, BBEH, etc.). Middle: GPQA subset. Right: GSM8K subset. The visualizations show clustering of different datasets or latent groups, revealing how different reasoning regimes occupy distinct regions in latent space.

Figure 8 shows heterogeneity both across benchmarks and within a benchmark. In the combined projection, MATH and GSM8K overlap strongly, as do portions of the QA-oriented groups, while other groups occupy different regions. A benchmark label therefore gives only a coarse view of the displayed representation. Within GPQA, the C0–C3 groups overlap extensively; within GSM8K, their projected organization is more separated. The two within-task panels demonstrate that variation remains after the task identity is fixed, which is the level at which a trajectory-conditioned policy makes its decisions.

The analytical connection to the core method is the distinction between a shared representation interface and the varied reasoning states it carries. The plot's cluster labels describe grouping in the displayed projection; the controller's operational use of trajectory information is measured by the shuffled-state intervention in Table 13. That intervention lowers both backbone macros by more than five points while retaining the same interface. Together, the visualization and intervention characterize representation structure and its use in control as two complementary empirical objects.

J.3 Out-of-distribution task profiles

Figure 9 shows that the strongest comparison method depends on the task. In its displayed evaluation, ReST-MCTS* leads the baselines on ARC-Challenge, BBH Unseen, HumanEval, and MBPP, but Best-of-N leads on Natural Questions at 41.5%. The full controller reaches 42.0% on Natural Questions, a 0.5-point advantage over that strongest row, compared with gains of 2.6, 3.2, 1.9, and 2.7 on the other tasks. The within-figure mean gain is approximately 2.2 points. This variation matters for the argument about allocation: a structured search baseline does not occupy the same relative position across all reasoning objectives.

Natural Questions also shows a reversal across baseline styles: ToT reaches 39.5%, below Greedy CoT at 40.2%, while sampling reaches 41.5%. In this evaluation, a more elaborate search can spend computation without improving the final answer. The smaller full-controller margin therefore identifies a task where the measured incremental opportunity is limited, alongside tasks with larger gains. The figure supplies these within-evaluation comparisons; Table 8 supplies the separately reported 15-task total-token matrix. Their common analytical question is how the available computation translates into task-standard output quality.

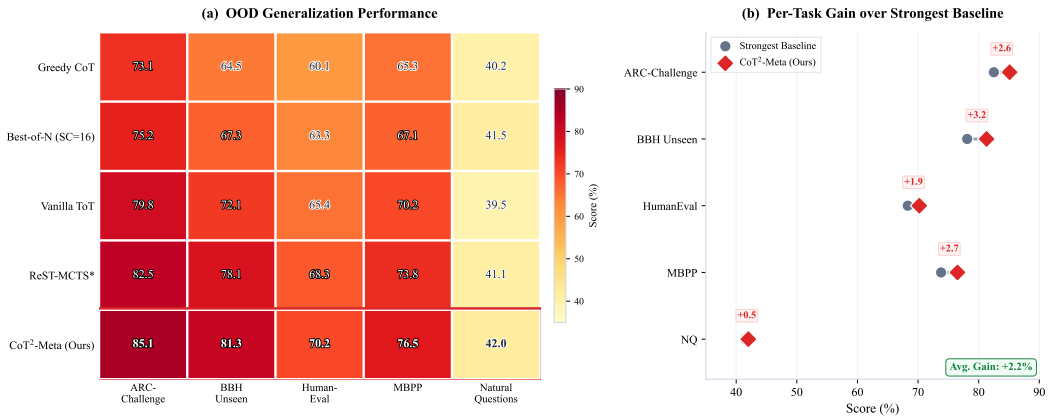


Figure 9: **OOD generalization across diverse benchmarks.** **(a)** Task-standard scores on ARC-Challenge, BBH Unseen, HumanEval, MBPP, and Natural Questions. **(b)** Per-task comparisons against the strongest baseline in the displayed evaluation.

J.4 Qualitative failure cases

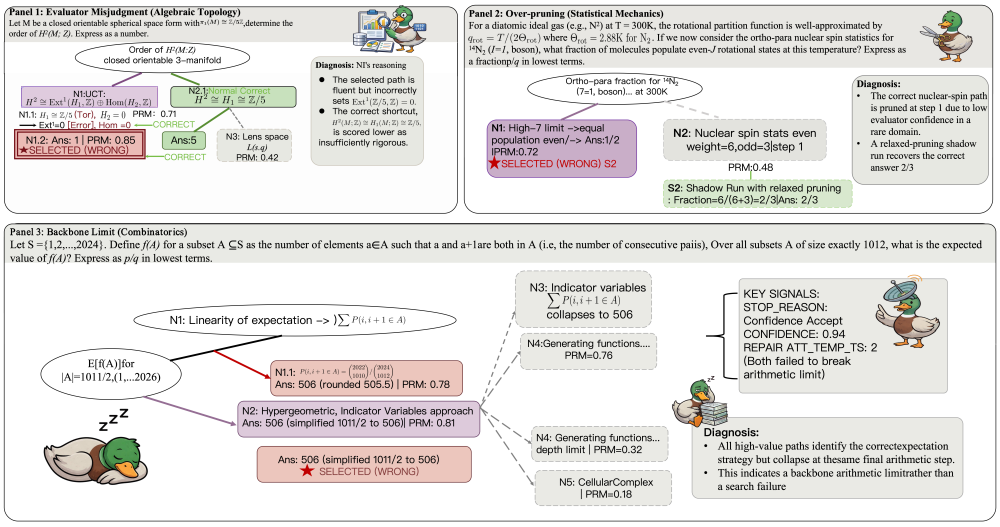


Figure 10: **Qualitative failure cases of CoT²-Meta on MATH.** Panel 1 shows evaluator misjudgment, where a fluent but flawed derivation is ranked above a simpler correct path. Panel 2 shows over-pruning, where a rare-domain argument is discarded early but recovered by a relaxed-pruning shadow run. Panel 3 shows a base-model limit, where all high-value branches identify the correct strategy yet collapse to the same final arithmetic error.

Figure 10 separates three points at which useful reasoning can be lost. In the evaluator-misjudgment example, a fluent flawed trajectory receives a stronger assessment than a simpler correct path. The candidate pool already contains an answer opportunity, so the operative failure is the ranking used to select it. In the over-pruning example, the relaxed-pruning shadow run recovers a path removed by the original controller. Here the important decision precedes final ranking: eliminating a partial trajectory prevents its later development. These cases give distinct meanings to evaluator quality and branch preservation, which appear as different audit categories in Table 27.

The final example shows several branches converging on the same arithmetic error after identifying a plausible strategy. Multiple branches can therefore share a defective continuation, so raw branch count and useful candidate diversity are different properties of the search. This connects the example to the effective-branch and duplication diagnostics in Table 24. The cases illustrate particular failure mechanisms; the designated-run counts in Table 27 quantify their broader audit categories. Their joint implication is that evaluation, retention, and continuation are separate intervention points inside one trajectory-management loop.

J.5 CONTROLLER EXECUTION TRACE

Algorithm 3: CoT^2-Meta executable controller trace

Input: problem x, token budget B, backbone G_θ
Initialize: root r, frontier $\mathcal{F} \leftarrow \{r\}$, ledger $C \leftarrow 0$
while $\mathcal{F} \neq \emptyset$ and $C < B$ **do**
  Refresh $m(n), v(n), U(n)$ for every $n \in \mathcal{F}$
  if a completed node has $v \geq 0.90$ **then** return the highest-v answer
  Compute $Q_{\text{exp}}, Q_{\text{rep}}, Q_{\text{prune}}$ for every feasible node–action pair
  $(n^\star, a^\star) \leftarrow \arg\max_{n,a} Q_a(n)$
  if $a^\star =$ EXPAND **then** choose Direct/Decompose/Verify by Eq. 8; generate and evaluate one child
  if $a^\star =$ REPAIR **then** preserve prefix, regenerate from $\tilde{i}$, and re-evaluate
  if $a^\star =$ PRUNE **then** remove $n^\star$ from $\mathcal{F}$
end while
Return the surviving completed candidate with highest v_{out} when $v_{\text{out}} \geq 0.70$; otherwise abstain.

Figure 11: Executable full-controller trace used by the headline CoT^2-Meta policy.

Figure 11 mirrors Algorithm 1 in compact form. The full controller refreshes the state of every active trajectory before each decision, compares node–action utilities across the frontier, and re-evaluates every generated or repaired trajectory before the next arbitration step. The threshold-only policy in Algorithm 2 shares the same execution operators and replaces frontier-wide arbitration with single-node selection followed by its fixed cascade.