REASONOPS: OPERATOR SEGMENTATION FOR LLM REASONING TRACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Chain-of-thought traces from large reasoning models can span tens of thousands of tokens, yet we lack a vocabulary for describing their internal structure. Previous methods developed to analyze chain-of-thought traces are either too rigid or not expressive enough, failing to capture features across domains and models. To remedy this, we introduce ReasonOps, an unsupervised method that discovers a shared vocabulary of discourse operators for segmenting and annotating chain-of-thought traces. Using ReasonOps, we analyze 44,662 traces from 12 thinking LLMs spanning 6 families across 8 reasoning benchmarks and discover that they share a common compositional structure: 7 recurring reasoning operators—discourse-level moves such as BACKTRACKING, INFERRING, and HYPOTHESIZING—that emerge from unsupervised clustering of sentence-initial 3-token pivots. These operators appear across every model family and benchmark domain, confirmed by three independent LLM judges who classify held-out samples at 70–76% accuracy. We analyze the structure of operators on easy vs. hard problems, revealing that the association between reflective operators and correctness changes with problem difficulty. Reasoning traces are highly model-identifying: structural operator features plus anchor-phrase text features recover the source model with macro-AUC $= 0.987$, revealing that each model family has a distinctive reasoning fingerprint. Structural operator features predict within-problem answer correctness well above baselines. Classifiers built on these operators reach WP-AUC $= 0.701$ globally and 0.801 on AIME. ReasonOps further enables early quality estimation well before the trace completes: we predict at WP-AUC $= 0.664$ for only 50% of the trace. Comparing OLMo checkpoints with a common base shows distinct operator distributions across post-training recipes, with greater backtracking under math- and code-focused training. The ReasonOps pipeline is unsupervised and annotation-free, enabling deep insights into LLM reasoning traces as well as strong downstream results on model identification and correctness prediction.

1 INTRODUCTION

Reasoning-capable large language models (LLMs) increasingly solve difficult problems by producing long intermediate traces before emitting a final answer. LLMs have become synonymous with "large reasoning models" (LRMs) as the majority of frontier models today are post-trained to elicit extended chain-of-thought reasoning before producing a final answer Yang et al. (2025a); Team et al. (2026); Guo et al. (2025); Agarwal et al. (2025). Prompting methods, multi-path decoding, process supervision, and reinforcement-learning-based post-training have all improved performance on mathematics, science, and coding benchmarks, but they have also made reasoning traces longer, more expensive, and harder to analyze Wei et al. (2022); Lightman et al. (2023); Guo et al. (2025). This comes at an increasing demand for monitoring and oversight of LLM decision-making processes Guan et al. (2025); Korbak et al. (2025); Bowman et al. (2022).

Since the advent of chain-of-thought (CoT) prompting Wei et al. (2022), reasoning traces have been hypothesized as a window into the problem-solving capabilities of LLMs. Reasoning traces, or chain-of-thought traces, are valuable artifacts for black-box understanding of LLMs as they can be obtained without access to model weights. These traces need not faithfully reflect every aspect of the underlying computation. These traces have been shown to contain rich information Shojaee et al. (2025); Kim et al. (2026), yet we lack a uniform vocabulary under which to characterize reasoning traces across models, domains, and datasets. While some methods have attempted to provide systematic annotations for reasoning traces, these methods often rely on *a priori* vocabularies and

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.

syntactic structures that are overfit to current models or domains Yang et al. (2025b); Venhoff et al. (2025); Lee et al. (2025); Bogdan et al. (2025); Li et al. (2025). In this paper, we ask: can we compress diverse traces into a small vocabulary of reasoning operators shared across model families, tasks, and domains while preserving predictive information about success and failure?

Our contributions. We introduce ReasonOps, an unsupervised framework for inducing a compact vocabulary of reasoning operators from visible chain-of-thought traces. We use sentence-initial 3-token pivots, frequency filtering, and semantic clustering with e5-small (Wang et al., 2022). We show that the resulting 7 operators are quantitatively meaningful: they generalize across 12 thinking LLMs from 6 families and 8 benchmarks, confirmed by three independent LLM judges (70–76% classification accuracy; chance: 14%). Structural operator features plus anchor-phrase text features identify the source model with macro-AUC $= 0.987$, revealing model-family reasoning fingerprints. Two human annotators recover 62.4% and 52.0% of induced labels on 250 shared held-out spans. For correctness prediction, we compare a gradient-boosted tree model using operator statistics and anchor text (Op-XGB) with an Operator Sequence Transformer (OST) that reads only operator labels. Both reach 0.701 WP-AUC under cross-dataset training, above the scalar baselines and LLM self-judge. Trained once on full traces, OST enables correctness prediction from partial traces at any depth. It reaches 0.664 at 50% depth and matches or slightly exceeds depth-specific Op-XGB models from that point onward. We open-source our codebase as a resource for the community.[1]

2 Related Work

Annotating and analyzing reasoning traces. Deepseek-R1 creators noted that "Wait" phrases were emergent behavior associated with reasoning capability increases Guo et al. (2025). This and the interpretability of chain-of-thought traces inspired work into annotating and describing reasoning traces. However, previous works primarily rely on *a priori* vocabularies of reasoning Yang et al. (2025b); Venhoff et al. (2025); Kargupta et al. (2025), operate on arbitrary syntactic features such as sentences Lee et al. (2025); Bogdan et al. (2025), or rely on domain-specific vocabularies Li et al. (2025). ReasonOps is an unsupervised method that does not assume *a priori* syntactic features or a fixed vocabulary of annotations. Other works have included *ad hoc* methods to analyze reasoning traces for particular hypotheses Shojaee et al. (2025); Kim et al. (2026).

Monitoring and scalable oversight. The scalable oversight community has extensively explored ways to characterize LLM behavior Bowman et al. (2022); Kenton et al. (2024); Engels et al. (2026), and chain-of-thought monitoring has been explored as a prominent direction Korbak et al. (2025); Guan et al. (2025). This work is tangential to ReasonOps in terms of reasoning trace characterization. Some work has also shown that chain-of-thought can be an unfaithful representation of model behavior Lanham et al. (2023), but recent work has attempted to improve this in frontier models Guan et al. (2025); Paul et al. (2024). Our work occupies a complementary point: we treat visible traces as behaviorally meaningful artifacts and ask whether they admit a stable, annotation-free meso-scale abstraction.

Correctness prediction. Some methods have been proposed for early correctness prediction, including LLM-only methods Miao et al. (2024); Xiang et al. (2026), and mechanistic interpretability techniques Zhao et al. (2026). Similar to this line of work is that of process reward models Lightman et al. (2023) and outcome reward models Cobbe et al. (2021), including work on verifiers in test-time scaling Zhang et al. (2025); Hosseini et al. (2024). This work has been particularly prolific in mathematics problems Cobbe et al. (2021); Lightman et al. (2023); Wang et al. (2024a). Other work has explored test time scaling techniques for self-verification, including SelfCheck Miao et al. (2024) and others Wang et al. (2023); Brown et al. (2024); Snell et al. (2024).

3 ReasonOps

ReasonOps learns a phrase-to-operator dictionary, then applies it to segment and label new traces. We collect responses from 12 reasoning models across six families on MATH-500 (Hendrycks et al., 2021), GPQA Diamond (Rein et al., 2024), AIME, LiveCodeBench (Jain et al., 2024), HumanEval (Chen et al., 2021), MMLU-Pro (Wang et al., 2024b), ARC-Challenge (Clark et al., 2018), and BIG-

[1] Anonymized repository: `https://anonymous.4open.science/r/ReasonOps`

Bench Hard (Suzgun et al., 2023). Removing truncated and malformed responses leaves **44,662 traces**, approximately 3,720 per model (Appendices A and B).[2]

3.1 Discovering discourse pivots

Let $x = (s_1, \ldots, s_L)$ be a trace after sentence segmentation. For each capitalized sentence s_i, we take its first n words in lowercase as its *pivot*, $p_n(s_i)$. If a sentence has fewer than n words, we use the whole sentence. For sentences that are not capitalized, we set $p_n(s_i) = \varnothing$. We use $n = 3$ to capture short discourse cues while limiting dependence on domain-specific content.

We retain pivots that recur across traces and datasets. Let $f(p)$ be the number of distinct traces containing pivot p, $d(p)$ the number of datasets containing it, and V_v the v most frequent corpus tokens. With $f_{\min} = 100$, $d_{\min} = 3$, and $v = 2000$, the retained vocabulary is

$$\mathcal{P} = \{p : \ f(p) \geq f_{\min},\ d(p) \geq d_{\min},\ \text{tokens}(p) \subseteq V_v\}\,. \tag{1}$$

These filters remove rare and domain-specific openings, leaving 5,464 pivots.[3]

We embed each retained pivot with the pretrained sentence encoder `e5-small-v2` (Wang et al., 2022). Writing its normalized embedding as $e(p)$, we fit K-means with 30 restarts:

$$\min_{c,\,\mu_1,\ldots,\mu_K} \sum_{p \in \mathcal{P}} \left\| e(p) - \mu_{c(p)} \right\|_2^2 . \tag{2}$$

Here $c(p) \in \{1, \ldots, K\}$ is the cluster assigned to pivot p, and μ_k is the centroid of cluster k. We name each cluster from its representative pivots and example spans. We select K by asking Claude Sonnet 4.6 to assign held-out spans to the induced categories from their descriptions, without seeing their assigned labels. We choose the value with the highest chance-corrected agreement (Cohen's κ): $K = 7$ achieves $\kappa = 0.693$ over $K \in \{6, \ldots, 11\}$. Section 4.1 describes semantic validation; Appendix C reports the sweep. Selection uses semantic agreement rather than downstream prediction performance (Appendix Q).

3.2 Segmenting and labeling a trace

A retained pivot starts a labeled span. We also start a span at (i) the first sentence of a trace, (ii) each bulleted or numbered list item, and (iii) a question following a non-question. Let $1 = b_1 < \cdots < b_M$ denote the resulting sentence indices and set $b_{M+1} = L + 1$. Let $a_j = p_3(s_{b_j})$ be the pivot of span j's first sentence. Here p_3 extracts the first three words in lowercase. Span j and its label are, for $j = 1, \ldots, M$,

$$S_j = (s_{b_j}, \ldots, s_{b_{j+1}-1}), \qquad \ell_j = \begin{cases} c(a_j), & a_j \in \mathcal{P}, \\ \text{unlabeled}, & \text{otherwise.} \end{cases} \tag{3}$$

The first sentence s_{b_j} is the span's *anchor*. A lexical match takes precedence over a structural cue. Structural spans without a retained pivot remain unlabeled (Appendix O). Removing these unlabeled entries gives an operator sequence $o(x) = (o_1, \ldots, o_T)$ for downstream analysis. Only the leading pivot determines the label: "I need to" maps to GROUNDING, but "Okay, so I need to" begins with "okay so i" and maps to INFERRING. The label describes the opening move rather than every clause (Appendix J).

Computational cost. Vocabulary discovery takes under five minutes for the corpus. Once the dictionary is fixed, annotation requires no embedding model or LLM call. It adds less than 1 ms per trace and takes under two minutes for the full 44K-trace corpus.

4 The Learned Operators

Each operator occurs in every model and dataset. They capture shared discourse functions, not an exhaustive inventory of cognitive operations. We group QUALIFYING, HYPOTHESIZING, and

[2] We use a 65,536-token context budget. Models are queried through OpenRouter, except Claude, which is queried through the Anthropic API with extended thinking.

[3] During development, lowering $f_{\min}$ to 50 admitted many domain-specific phrases. Varying v from 1,500 to 3,000 did not qualitatively change the clusters.

BACKTRACKING as *reflective*; INITIATING, GROUNDING, INFERRING, and CONSTRAINING are *committal*. These descriptive groups do not alter segmentation. Coincidentally, these operators align roughly with Schoenfeld's Episode Theory (Schoenfeld, 1985): INFERRING resembles Implement, HYPOTHESIZING Explore, and BACKTRACKING Monitor/Verify. The seven operators capture recurring discourse moves, while some reasoning behaviors remain unlabeled or share broader labels (Appendix O).

Table 1: The seven learned operators, their characteristic pivots, and their interpretations. A short sentence may supply fewer than three tokens.

Operator	Representative Pivots	Description
INITIATING	let me think, let me check, let me verify	Signals the start of a new line of reasoning
QUALIFYING	hmm, but the problem, but let me	Introduces caveats or complications
GROUNDING	i need to, the question is, looking at the	Anchors reasoning in facts or given information
INFERRING	so the answer, thus final answer, yes	Draws a conclusion from prior steps
HYPOTHESIZING	alternatively perhaps the, for example if, so perhaps the	Entertains a tentative or conditional scenario
BACKTRACKING	wait, wait no, wait let me	Signals potential error and prepares to restart
CONSTRAINING	we need to, now we need, so we need	Identifies necessary conditions, narrows solution space

4.1 SEMANTIC VALIDATION

We evaluate the operators through label recovery and naming stability. Label recovery measures agreement between the induced assignments and classifications made by LLM judges or human annotators. Naming stability measures the consistency of cluster descriptions across repeated samples of representative spans.

Label recovery. The evaluation set contains $N = 350$ held-out spans, with 50 from each operator. For each span S_i, a judge predicts an operator $\hat{y}_i$ using a codebook of operator names, definitions, and twelve representative pivots per operator. We compare $\hat{y}_i$ with the induced label y_i, which is withheld from the judge, and report classification accuracy and Cohen's κ. We evaluate three LLM judges with this protocol. Their baseline prompts contain no example spans; a second condition supplies Claude Sonnet 4.6 with three examples per operator. Human agreement is measured on the 250 spans labeled by both annotators.

Table 2: LLM-judge validation (n=350 spans). Baseline prompts include names, descriptions, and 12 pivots; exemplars add 3 spans per operator.

Judge	Prompt variant	Accuracy
Random baseline	—	14.3%
Claude Haiku 4.5	baseline	76.0%
GPT-5.4-Mini	baseline	71.7%
Claude Sonnet 4.6	baseline	69.7%
Claude Sonnet 4.6	exemplars	**74.6%**

Accuracy is 69.7–76.0%, or 4.9–5.3 times chance; examples improve Sonnet from 69.7% to 74.6% (Table 2). On 250 shared spans, two human annotators recover 62.4% and 52.0% of induced labels ($\kappa = 0.562$ and 0.443). Inter-annotator agreement is 58.0% ($\kappa = 0.504$). Both recover INITIATING and BACKTRACKING most consistently (Appendix N).

Naming stability. For each fixed cluster k, we repeat naming $R = 30$ times with twelve fixed pivots and five resampled spans per prompt. Let $v_{k,r}$ be the unit-normalized embedding of the description from repetition r. Its similarity to repetition s is $\rho_{k,r,s} = v_{k,r}^{\top} v_{k,s}$, yielding $\binom{30}{2} = 435$ pairwise cosine similarities per cluster. Comparisons use descriptions from different clusters and from randomly grouped spans. Median similarity is 0.660 within clusters, versus 0.467 across clusters and 0.565 for random groups. All seven clusters show greater within-cluster similarity. This tests description stability with cluster membership fixed (Appendix K.1).

5 ANALYSIS OF OPERATOR USAGE

We compare operator usage across models, post-training recipes, and answer outcomes. These observational comparisons do not establish causal effects on accuracy.

Operator	AIME	MATH	GPQA	ARC-C
INITIATING	Let me be more systematic about tracking the solutions.	Let me also verify by finding the side length directly.	Let me check if a 4-membered ring is actually feasible.	Let me confirm: kinetic energy is the energy of motion.
QUALIFYING	But if S is all rows or T is all columns, no colorings.	But the phrase "2 people on a side" might be ambiguous.	But if X is not Hermitian, then e^X may not be unitary.	But the answer might be based on the given information.
GROUNDING	I need to find the sum of the squares of all the items.	I need to convert the decimal number 999 into base six.	I need to work through the bridge structure more carefully.	I need to determine which is the first line of defense.
INFERRING	So (11,4,4) is a solution, so x=11 is a possible value.	So (1 - i)^8 is indeed 16, so the absolute value is 16.	So A is the intrinsic angular momentum operator (spin).	So D is a direct effect, while C is an indirect effect.
HYPOTHESIZING	Alternatively, another approach is to use substitution.	Alternatively, I can use modular arithmetic properties.	Alternatively, consider the reaction of O2 with metals.	Alternatively, maybe the answer is (D) flat, wide tail.
BACKTRACKING	But wait, another thought: Is 155 the minimal solution?	But wait, I need to check if 13 is actually achievable.	But wait, I should double-check whether e^X is unitary.	But wait, is a jar appropriate for monarch butterflies?
CONSTRAINING	Now we need intersection of angle bisectors at A and D.	Now we need average price (in cents) of his 70s stamps.	So we need to add NaOH to provide 0.00025 moles of OH-.	So we need to test the design's stability and buoyancy.

Figure 1: Examples of all seven operators across four benchmarks. Each cell is a verbatim anchor with its induced label. Labels describe the leading discourse cue, not every function in the sentence. Additional benchmark examples and a complete trace appear in Appendix J.

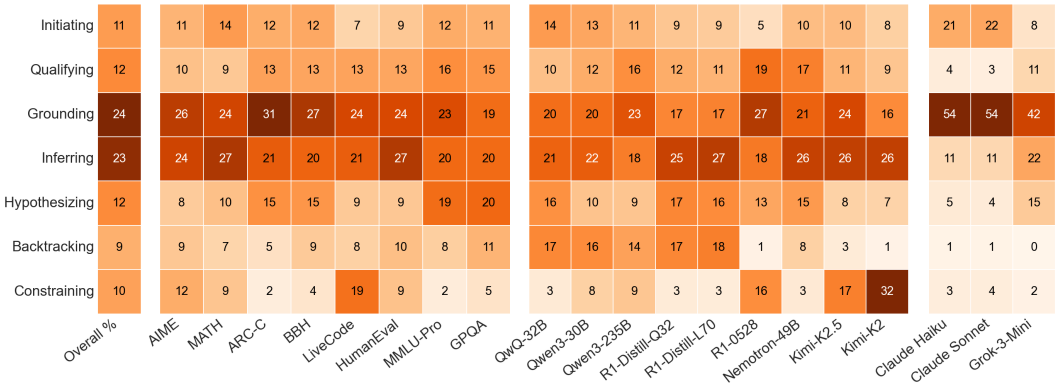


Figure 2: Operator usage (%) overall, by dataset, and by model. Rows are operators; cells show each operator's share of labeled spans.

Operator frequencies vary across datasets and model families (Figure 2). Mathematical traces favor GROUNDING and CONSTRAINING; GPQA and MMLU-Pro use more HYPOTHESIZING and QUALIFYING. Coding traces show elevated CONSTRAINING. The R1-distilled and Qwen models resemble their respective relatives; Kimi favors CONSTRAINING, and Claude and Grok favor GROUNDING.

5.1 OPERATOR DISTRIBUTION REFLECTS POST-TRAINING STRATEGY

Operator profiles change substantially across post-training recipes even when architecture and base model are held fixed (Figure 3). We compare three 7B OLMo RL-Zero checkpoints (Allen Institute for AI, 2025): General, trained on open-ended chat prompts with an LM-judge reward; Math, trained with rule-based verification; and Code, trained with execution-based verification.

On both AIME and LiveCodeBench, Math and Code use more INFERRING and BACKTRACKING and less GROUNDING and HYPOTHESIZING than General.

The profile persists when Math is evaluated on code and Code on mathematics, showing that operator fingerprints reflect post-training as well as architecture and current task. Because the variants differ in both training queries and reward mechanism, this experiment isolates the combined post-training recipe rather than either component alone (Appendix P).

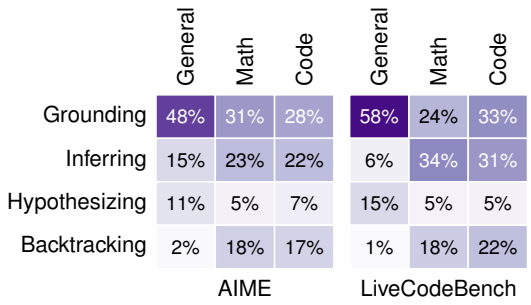


Figure 3: OLMo General, Math, and Code variants operator usage (%) across 254 traces; darker = higher usage.

5.2 Operator usage on correct vs. incorrect traces

Committal operators seem to be overrepresented in correct traces relative to incorrect traces, with Grounding, Inferring, Initiating, and Constraining all being positive in correct - incorrect usage (Figure 4).

Conversely, reflective operators are globally seen more often in incorrect traces than correct traces. Hypothesizing and Qualifying show particularly higher usage in incorrect traces than correct traces.

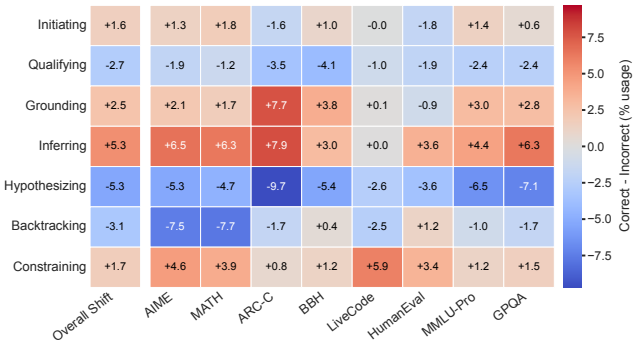


Figure 4: Correct minus incorrect operator usage (percentage points). Red denotes higher usage in correct traces; blue denotes lower usage.

Easy vs. hard problems. We next define a problem as "hard" if fewer than 1/3 of models solve it, and "easy" if more than 2/3 models pass. We first filter to only BBH, LiveCodeBench, MMLU-Pro, and GPQA, which have sufficient samples of hard problems. We define the *committal-reflective gap* as the difference in usage between the two groups. On easy problems, this gap is much higher in correct traces ($+44.2\%$) than in incorrect traces ($+7.5\%$) and is strongly predictive of correctness (AUC = 0.76 $[0.74, 0.78]$ 95% CI).

The committal-reflective gap flips, showing $+30.4\%$ on correct traces and $+34.9\%$ on incorrect traces, a -4.5% shift which is not statistically-significant (Mann-Whitney p=0.15) (Figure 5a). The predictive signal disappears as the AUC for correct vs. incorrect via committal-reflective gap alone drops to AUC = 0.48 $[0.44, 0.51]$. However, examining Hypothesizing vs. Inferring, instead of all committal and reflective operators, reveals a difference. This gap increases monotonically over the length of the trace, reaching 6.6% at the tail of the trace, with incorrect traces showing much higher usage of Inferring over Hypothesizing (Figure 5b). On easy problems, reflection marks failure; on hard problems, late Hypothesizing marks success.

5.3 Backtracking is a local operator

Both operators semantically appear to interrupt a model's current line of reasoning, but we asked whether these operators actually reflected global strategic shifts in the model's problem-solving. We used Claude Sonnet 4.6 as an LLM judge to classify a stratified sample of GPQA events as Local, Sub-Problem, or Global (Appendix H). **Backtracking and Hypothesizing are overwhelmingly local revisions rather than strategy changes**: 85.4% of backtracking and 80.2% of hypothesizing events are Local, while only 1.6% of either are Global; their scope distributions are statistically indistinguishable. This is supported by literature that shows that backtracking is often superficial Wang et al. (2026) and rarely changes a model's answer Kang et al. (2025).

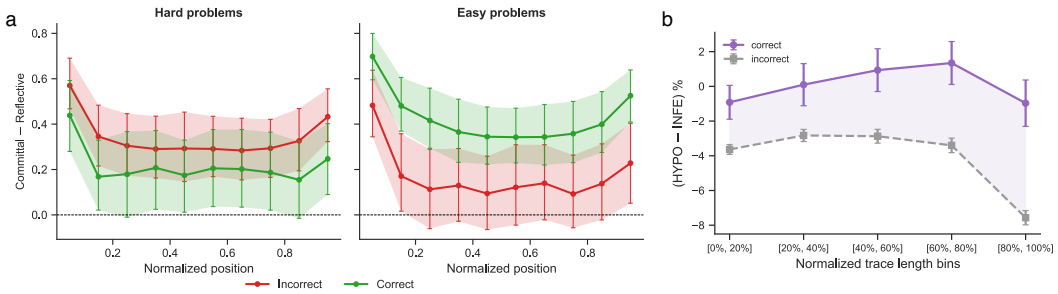


Figure 5: Operator usage over trace position. a) Committal-minus-reflective gap on hard and easy problems. b) Hypothesizing-minus-inferring gap on hard problems. 95% CI shown.

6 Predicting Trace Source and Outcome

We predict source-model identity and answer correctness with a fixed operator vocabulary. Cross-validation assigns all attempts at a problem to the same fold. We compare a gradient-boosted tree classifier using operator statistics and anchor text with a Transformer using only operator labels.

Operator-feature classifier. For operator sequence $o(x)$, let $\phi(o(x)) \in \mathbb{R}^{117}$ contain operator frequencies, position-dependent frequencies, transitions, run lengths, and sequence summaries. Let $\psi(x) \in \mathbb{R}^D$, $D \leq 8000$, be a term-frequency–inverse-document-frequency (TF-IDF) vector of anchor text, with its vocabulary fitted within each training fold. We concatenate these features and train an XGBoost classifier (Chen & Guestrin, 2016):

$$z(x) = \big[\phi(o(x)); \psi(x)\big] \in \mathbb{R}^{117+D}. \tag{4}$$

We call this classifier *Op-XGB*: 400 trees of depth six, trained for multiclass model identification or binary correctness (Appendix E).

Operator sequence model. To test the information in labels alone, we train an *Operator Sequence Transformer* (OST). Each position receives an operator embedding, a transition embedding for the preceding and current operators, and a sinusoidal encoding of its normalized position. A four-layer, four-head Transformer with hidden width 128 encodes the sequence. Attention pooling and a linear prediction head produce a scalar score $f_\theta(o(x))$; the model has approximately 800,000 parameters.

The training objective ranks correct attempts above incorrect attempts at the same problem. If $\mathcal{B}$ is a batch of such pairs (x^+, x^-), its loss is

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{(x^+,x^-)\in\mathcal{B}} \log\big(1 + \exp\big[f_\theta(o(x^-)) - f_\theta(o(x^+))\big]\big)\,. \tag{5}$$

Training uses full sequences; inference can use prefixes with positions renormalized to the visible length (Appendix D).

6.1 Model Identification

We train the multiclass Op-XGB predictor to distinguish the twelve source models from trace alone. Under five-fold problem-level cross-validation, it achieves 79.9% accuracy, compared with 8.3% chance accuracy, and a macro-AUC of 0.987. Per-model AUCs range from 0.967 for R1-Distill-Llama-70B to 0.997 for Kimi-K2, Kimi-K2.5, and Grok-3-Mini (Figure 6).

Errors concentrate among the R1-distilled, Claude, and Kimi model pairs (Appendix L.2). On 1,470 new traces, persona and evasion prompts retain 83–99% identification accuracy.

Mimicking another model's openers reduces Qwen3-30B to 60%, while Kimi-K2.5 and R1-0528 remain at 94%. These results hold under adversarial prompting that attempts to change model identity via trace construction (Appendix R).

6.2 CORRECTNESS PREDICTION

We evaluate correctness on six benchmarks: AIME, ARC-Challenge, GPQA, LiveCodeBench, MATH-500, and MMLU-Pro. BIG-Bench Hard and HumanEval are excluded from this comparison for the grading and task considerations (Appendix B). We report *within-problem AUC* (WP-AUC), which averages discrimination between correct and incorrect attempts at the same problem, controlling for differences in problem difficulty (Zhou et al., 2018). The formal definition is in Appendix M.

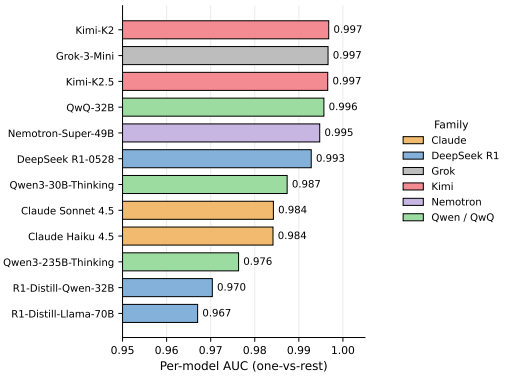


Figure 6: Op-XGB model-identification AUC under problem-level cross-validation. Dashed line: macro-AUC 0.987. Axis starts at 0.95.

We use five-fold problem-level cross-validation in two settings. Cross-dataset (CD) training pools the six datasets within each fold; in-dataset (ID) training fits a separate predictor for each benchmark. CD does not hold out entire datasets. We compare both learned predictors with logistic regression on three scalar features—log trace length, backtracking frequency, and "wait" count—and with SelfCheck, in which the generating model judges its own reasoning without access to the correct answer (Miao et al., 2024) (Appendix M).

Table 3: Correctness WP-AUC under five-fold problem-level cross-validation. CD pools datasets; ID trains per dataset. Shading marks our methods; bold/underlining mark the best/second-best scores. OST uses full sequences (Appendices E, M).

	Op-XGB (ours)		OST (ours)		Length		Backtrack		Wait		SelfCheck	
Dataset	CD	ID	CD	ID	CD	ID	CD	ID	CD	ID	CD	ID
AIME	0.779	**0.838**	0.801	0.797	0.500	0.500	0.621	0.621	0.679	0.679	0.501	0.501
ARC-Challenge	0.618	0.559	**0.645**	0.556	0.559	0.559	0.520	0.520	0.501	0.499	0.578	0.578
GPQA	0.682	**0.703**	0.691	0.691	0.591	0.591	0.591	0.591	0.602	0.602	0.526	0.526
LiveCodeBench	0.754	**0.795**	0.727	0.732	0.426	0.574	0.616	0.616	0.591	0.591	0.496	0.496
MATH	0.597	0.570	0.612	**0.662**	0.407	0.407	0.464	0.464	0.483	0.517	0.504	0.504
MMLU-Pro	0.635	**0.639**	0.635	0.638	0.587	0.587	0.628	0.628	0.633	0.633	0.520	0.520
Global	0.701	**0.723**	0.701	0.703	0.507	0.551	0.594	0.594	0.600	0.603	0.512	0.512

Both predictors outperform the baselines overall (Table 3). Op-XGB reaches 0.701/0.723 WP-AUC under CD/ID training; OST reaches 0.701/0.703 without anchor text. The strongest scalar baseline reaches 0.603, and SelfCheck is near chance at 0.512. Neither learned predictor dominates every benchmark (Appendix I).

6.3 EARLY CORRECTNESS PREDICTION

We test whether correctness can be predicted before a trace is complete. We evaluate the OST on the first 10%, 25%, 50%, 75%, and 100% of each operator sequence, using the model trained on full traces at every depth. As a depth-specific comparison, we recompute the operator and anchor-text features and train a separate XGBoost classifier at each depth; we call this variant *Op-XGB-Early*. Partial-trace SelfCheck receives the same fraction of the original reasoning text.

OST improves monotonically from 0.605 WP-AUC at 10% depth to 0.664 at 50% and 0.701 at completion. On AIME, the corresponding scores are 0.645, 0.740, and 0.801; LiveCodeBench and GPQA reach 0.727 and 0.691 at completion. From 50% onward, OST matches or slightly exceeds separately trained Op-XGB-Early models; partial SelfCheck remains near chance at 0.50–0.56 (Appendix M).

We also found that this approach can slightly improve test-time scaling through correctness prediction. A separate operator-feature scorer improves AIME best-of-N accuracy by 2.4–4.0 points

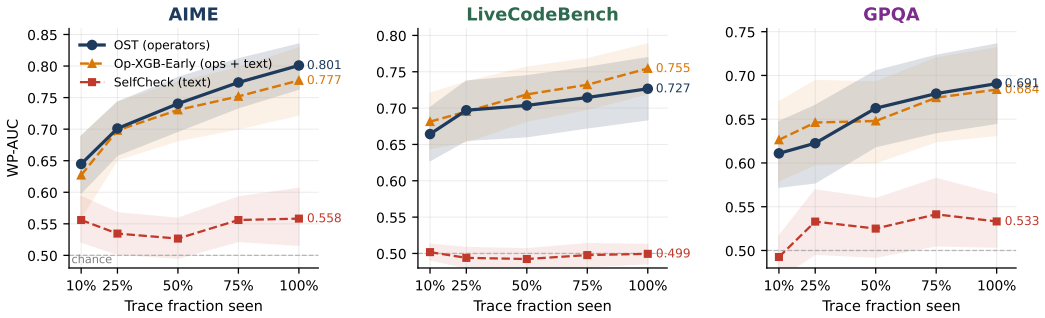


Figure 7: Correctness prediction from partial traces. OST is trained once on full operator sequences, while Op-XGB-Early is fitted separately at each depth. SelfCheck uses the generating model's judgment of its partial reasoning. Shaded bands give 95% confidence intervals.

over random selection at matched generation spend, but does not significantly beat trace length. OST-guided abort-and-restart does not improve accuracy (Appendix S).

7 Discussion

ReasonOps is an unsupervised method for annotating reasoning traces, built from careful observations of their semantic structure. ReasonOps is a bottom-up method that, rather than applying predefined cognitive behaviors to traces Li et al. (2025); Gandhi et al. (2025), induces operators from recurring language, since fixed taxonomies can over-constrain the strategies of reasoning models Lee et al. (2026). Clause-initial cue phrases are established signals of discourse structure Litman (1996); Zeldes et al. (2025), and local lexical n-grams traditionally suffice to identify discourse-unit boundaries Sediqin & Argamon (2025). Three-token pivots strike a useful balance for discourse analysis: they disambiguate the polysemy of single-token markers like "but" or "so" while remaining short enough to recur across tens of thousands of traces. The top-2,000 vocabulary filter is essential to this approach, as it excludes high-frequency but domain-specific phrases that would otherwise contaminate the pivot set with non-discourse-functional content. The sensitivity analysis supports these choices, and $K = 7$ maximizes independent judge agreement among the tested operator counts (Appendix Q).

ReasonOps provides compact annotations that transfer across the models and domains studied here, enabling comparative analysis and downstream tasks such as model identification and correctness prediction. Further, CoT traces need not faithfully reflect a model's underlying computation (Lanham et al., 2023; Chen et al., 2025), but ReasonOps does not require this assumption. Operators summarize the structure of emitted traces and predict observable outcomes such as correctness and source model. Operator labels should therefore be read as discourse moves in the text rather than as the model's internal reasoning steps.

Future work. We see several avenues for future work on annotating reasoning traces. An extension could be made to agentic traces, i.e., traces of actions taken by an LLM when enabled with a harness. The ReasonOps approach, which is unsupervised and does not assume *a priori* sets of operators, might be useful for agentic traces where actions may change drastically based on the harness. OST's early correctness signal also motivates further study of compute-adaptive generation. Operator-based selection improves over random selection at matched generation spend in our experiments; developing effective online intervention policies remains a direction for future work. Controlled post-training studies could further connect operator distributions to training data and reward design.

Broader impacts. This work paves the way to a common language for describing reasoning traces of LLMs. This could be used for further analysis of LLMs as well as downstream tasks beyond correctness prediction.

AI USE STATEMENT

LLM tools were used for editing and drafting text, understanding technical concepts, implementing standard methods, data processing and filtering, facilitating experiments, and preparing visualizations. LLMs also serve as experimental judges and self-check baselines, as described in the methods and appendix.

ETHICS STATEMENT

Our work did not involve the use of human subjects or of sensitive data. Our datasets are fully available as our data is not sensitive. We do not believe that this work poses any dangerous or harmful insights. On the contrary, we believe our work could be used to further enable understanding and comparison of chain-of-thought readouts from language models to improve safety.

REPRODUCIBILITY STATEMENT

We have included a link to an anonymized GitHub in our submission. We took great lengths to ensure that our submitted codebase contains code that can be used to reproduce the results shown in this paper. In addition, our appendix details technical decisions and more detail needed to reproduce experiments.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Allen Institute for AI. Olmo 3 RL-Zero model cards, 2025. URL `https://huggingface.co/allenai/Olmo-3.1-7B-RL-Zero-Math`.

Paul C Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. Thought anchors: Which llm reasoning steps matter? *arXiv preprint arXiv:2506.19143*, 2025.

Samuel R Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, Andy Jones, Anna Chen, et al. Measuring progress on scalable oversight for large language models. *arXiv preprint arXiv:2211.03540*, 2022.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, et al. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025.

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. *arXiv preprint arXiv:2110.14168*, 2021.

Joshua Engels, David D. Baek, Subhash Kantamneni, and Max Tegmark. Scaling laws for scalable oversight. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=u1j6RqH8nM.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STaRs. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2503.01307.

Melody Y Guan, Miles Wang, Micah Carroll, Zehao Dou, Annie Y Wei, Marcus Williams, Benjamin Arnav, Joost Huizinga, Ian Kivlichan, Mia Glaese, et al. Monitoring monitorability. *arXiv preprint arXiv:2512.18311*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

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. *NeurIPS*, 2021.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-STar: Training verifiers for self-taught reasoners. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=stmqBSW2dV.

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.

Liwei Kang, Yue Deng, Yao Xiao, Zhanfeng Mo, Wee Sun Lee, and Lidong Bing. First try matters: Revisiting the role of reflection in reasoning models. *arXiv preprint arXiv:2510.08308*, 2025.

Priyanka Kargupta, Shuyue Stella Li, Haocheng Wang, Jinu Lee, Shan Chen, Orevaoghene Ahia, Dean Light, Thomas L Griffiths, Max Kleiman-Weiner, Jiawei Han, et al. Cognitive foundations for reasoning and their manifestation in llms. *arXiv preprint arXiv:2511.16660*, 2025.

Zachary Kenton, Noah Y Siegel, János Kramár, Jonah Brown-Cohen, Samuel Albanie, Jannis Bulian, Rishabh Agarwal, David Lindner, Yunhao Tang, Noah D Goodman, et al. On scalable oversight with weak llms judging strong llms. *Advances in Neural Information Processing Systems*, 37:75229–75276, 2024.

Junsol Kim, Shiyang Lai, Nino Scherrer, James Evans, et al. Reasoning models generate societies of thought. *arXiv preprint arXiv:2601.10825*, 2026.

Tomek Korbak, Mikita Balesni, Elizabeth Barnes, Yoshua Bengio, Joe Benton, Joseph Bloom, Mark Chen, Alan Cooney, Allan Dafoe, Anca Dragan, et al. Chain of thought monitorability: A new and fragile opportunity for ai safety. *arXiv preprint arXiv:2507.11473*, 2025.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Jinu Lee, Sagnik Mukherjee, Dilek Hakkani-Tur, and Julia Hockenmaier. Reasoningflow: Semantic structure of complex reasoning traces. *arXiv preprint arXiv:2506.02532*, 2025.

Seongyun Lee, Seungone Kim, Minju Seo, Yongrae Jo, Dongyoung Go, Hyeonbin Hwang, Jinho Park, Xiang Yue, Sean Welleck, Graham Neubig, Moontae Lee, and Minjoon Seo. The CoT encyclopedia: Analyzing, predicting, and controlling how a reasoning model will think. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ugZKZ8vufv.

Ming Li, Nan Zhang, Chenrui Fan, Hong Jiao, Yanbin Fu, Sydney Peters, Qingshu Xu, Robert Lissitz, and Tianyi Zhou. Understanding the thinking process of reasoning models: A perspective from schoenfeld's episode theory. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 18278–18299, 2025.

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.

Diane J. Litman. Cue phrase classification using machine learning. *Journal of Artificial Intelligence Research*, 5:53–94, 1996. doi: 10.1613/jair.327. URL https://arxiv.org/abs/cs/9609102.

Ning Miao, Yee Whye Teh, and Tom Rainforth. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=pTHfApDakA.

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

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. *arXiv preprint arXiv:2311.12022*, 2024.

Alan H. Schoenfeld. *Mathematical Problem Solving*. Academic Press, Orlando, FL, 1985. ISBN 9780126288704.

Mohammadreza Sediqin and Shlomo Engelson Argamon. ESURF: Simple and effective EDU segmentation. *arXiv preprint arXiv:2501.07723*, 2025. URL https://arxiv.org/abs/2501.07723.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *arXiv preprint arXiv:2506.06941*, 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.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. Understanding reasoning in thinking language models via steering vectors. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=OwhVWNOBcz.

Hanbin Wang, Jingwei Song, Jinpeng Li, Qi Zhu, Fei Mi, Ganqu Cui, Yasheng Wang, and Lifeng Shang. Teaching large reasoning models effective reflection. *arXiv preprint arXiv:2601.12720*, 2026.

Liang Wang, Nan Yang, Xinlong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

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, 2024b.

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.

Yang Xiang, Yixin Ji, Ruotao Xu, Dan Qiao, Zheming Yang, Juntao Li, and Min Zhang. When is thinking enough? early exit via sufficiency assessment for efficient reasoning. *arXiv preprint arXiv:2604.06787*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Shiming Yang, Yuxuan Tong, Xinyao Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning. In *Forty-second International Conference on Machine Learning*, 2025b.

Amir Zeldes, Tatsuya Aoyama, Yang Janet Liu, Siyao Peng, Debopam Das, and Luke Gessler. eRST: A signaled graph theory of discourse relations and organization. *Computational Linguistics*, 51 (1):23–72, 2025. doi: 10.1162/coli_a_00538. URL https://aclanthology.org/2025.cl-1.3/.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ccwp4tFEtE.

Zheng Zhao, Yeskendir Koishekenov, Xianjun Yang, Naila Murray, and Nicola Cancedda. Verifying chain-of-thought reasoning via its computational graph. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=CxiNICq0Rr.

Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1059–1068, 2018.

A Model Details

Table 4 lists the 12 thinking models used for trace collection. All expose raw chain-of-thought tokens accessible via API. All models except Claude are queried via the OpenRouter API; reasoning tokens are returned in the `reasoning` field of the OpenRouter response message. The `reasoning` request parameter is set to `{effort: high}` for Grok-3-Mini, `{max_tokens: N}` for Qwen3 thinking variants, and omitted for DeepSeek, QwQ-32B, Kimi, and Nemotron models. Claude models are accessed directly via the Anthropic API with `thinking: {type: enabled, budget_tokens: N}` enabled; raw thinking content is extracted from `type: thinking` response blocks.

Table 4: Models used for trace collection. All 12 expose raw chain-of-thought tokens.

Model	Family	Size	Notes
QwQ-32B	Qwen	32B	(Qwen Team, 2025)
Qwen3-30B-Thinking	Qwen	30B (3B act.)	MoE (Yang et al., 2025)
Qwen3-235B-Thinking	Qwen	235B (22B act.)	MoE (Yang et al., 2025)
DeepSeek-R1-0528	DeepSeek	671B MoE	(DeepSeek-AI et al., 2025)
R1-Distill-LLaMA-70B	DeepSeek	70B	Distilled (DeepSeek-AI et al., 2025)
R1-Distill-Qwen-32B	DeepSeek	32B	Distilled (DeepSeek-AI et al., 2025)
Claude-Sonnet-4.5	Anthropic	—	Extended thinking (Anthropic, 2025)
Claude-Haiku-4.5	Anthropic	—	Extended thinking (Anthropic, 2025)
Grok-3-Mini	xAI	—	(xAI, 2025)
Nemotron-Super-49B	NVIDIA	49B	(Bercovich et al., 2025)
Kimi-K2-Thinking	Moonshot	MoE	(Kimi Team, 2025)
Kimi-K2.5	Moonshot	MoE	(Team et al., 2026)

B Corpus Statistics

The full corpus spans 8 benchmarks and contains 44,662 traces across 12 models. Table 5 reports per-dataset statistics for the 6-dataset correctness prediction subset (33,209 traces), which excludes BBH and HumanEval due to grading ambiguity or near-ceiling accuracy. Each problem was attempted by all 12 models with up to 5 independent samples per model; the AIME subset contains 90 problem IDs.

Table 5: Per-dataset trace counts in the 6-dataset correctness prediction corpus. All 12 models contribute to every dataset. Accuracy = fraction of traces judged correct by the official grader.

Dataset	Traces	Problems	Models	Accuracy
AIME	4,604	90	12	86.4%
ARC-Challenge	5,808	97	12	94.9%
GPQA Diamond	5,615	100	12	74.0%
LiveCodeBench	5,516	99	12	33.5%
MATH-500	5,794	100	12	94.6%
MMLU-Pro	5,872	100	12	76.3%
Total	**33,209**	**586**		

The full operator-discovery corpus additionally includes BIG-Bench Hard and HumanEval (44,662 traces total across all 8 benchmarks). These two datasets are excluded from correctness prediction: BBH uses answer-matching formats where operator structure shows near-chance predictive signal, and HumanEval correctness is determined by code execution rather than reasoning quality.

C K Selection

Table 6 reports LLM-judge κ (Claude Sonnet 4.6) for each K in the sweep. K=7 achieves the highest κ.

Table 6: K selection via LLM-judge κ (Claude Sonnet 4.6).

K	6	**7**	8	9	10	11
κ	0.604	**0.693**	0.611	0.648	0.573	0.558

D Operator Sequence Transformer (OST) Architecture

The Operator Sequence Transformer predicts correctness from the ordered labels produced by ReasonOps. Unlabeled structural spans are excluded. Typical sequences contain 50–500 operator tokens; the implementation represents the seven labels with indices 0–6.

Input representation. For a visible sequence of length T, the input at position t is the sum of three 128-dimensional vectors:

$$h_t^{(0)} = E_u[o_t] + E_b[o_{t-1}, o_t] + \mathrm{PE}(r_t), \tag{6}$$

where E_u embeds the current operator, E_b embeds the preceding transition, and r_t is the normalized position within the visible sequence. We set $r_t = (t-1)/\max(T-1, 1)$ for $t = 1, \ldots, T$, and use $o_0 = \mathrm{START}$ for the first transition. The sinusoidal encoding PE supplies position information without adding a learned position embedding for each sequence length.

Encoder and prediction head. A pre-LayerNorm Transformer processes these inputs with full attention. Its configuration is

$$(N_{\text{layers}},\, d_{\text{model}},\, N_{\text{heads}},\, d_{\text{FF}}) = (4, 128, 4, 512). \tag{7}$$

Attention pooling with a learned query reduces the final sequence representations to one 128-dimensional vector. The prediction head is a single linear layer, $128 \to 1$, producing a scalar correctness score. The complete model has approximately 800,000 parameters. Training applies 5% token dropout.

Optimization. Training pairs contain a correct and an incorrect attempt at the same problem. The pairwise objective in Equation 5 encourages the correct attempt to receive the higher score. We sample 16 pairs per problem with both correct and incorrect attempts, repeating examples when needed. The pairs are fixed for each training fold and shuffled each epoch. Optimization uses AdamW with learning rate 3×10^{-4}, weight decay 10^{-4}, and batches of 64 pairs (128 traces). Early stopping uses validation WP-AUC when at least ten validation problems contain both outcomes; otherwise, it uses pooled AUC.

Partial traces. The model is trained on complete operator sequences and evaluated on prefixes containing the first $p\%$ of labels. Position encodings are recomputed relative to the visible prefix length; the weights remain fixed. This permits comparison across trace depths without training a separate sequence model at each depth.

E Op-XGB Feature Details

The operator-feature classifier uses the representation $z(x)$ in Equation 4. Its two components have distinct roles: $\phi(o(x))$ describes the operator sequence, while $\psi(x)$ retains lexical information from anchor sentences. The latter includes words beyond the three-token pivots used for discovery.

Operator features. Let $o = (o_1, \ldots, o_T)$ be a trace's labeled sequence. Global operator frequencies and adjacent-pair frequencies are

$$u_k = \frac{1}{\max(T,1)} \sum_{t=1}^{T} \mathbf{1}[o_t = k], \qquad B_{ab} = \frac{1}{\max(T-1,1)} \sum_{t=2}^{T} \mathbf{1}[o_{t-1} = a,\ o_t = b]. \tag{8}$$

An empty sequence has zero frequencies; for $T < 2$, adjacent-pair features are zero. The complete feature vector has six components:

- Global frequencies contribute 7 features, one for each u_k.
- Frequencies within four contiguous sequence quartiles contribute $7 \times 4 = 28$ features.
- Five scalar summaries record frequency entropy, maximum frequency, the number of observed operators, $\log(1 + T)$, and the fraction of adjacent labels that repeat.
- One-hot encodings of the first and last labels contribute $7 + 7 = 14$ features.
- Flattening the transition matrix B contributes 49 features.
- The mean and maximum run length of each operator, normalized by T, contribute another $7 + 7 = 14$ features.

These components sum to 117 features and use no trace text.

Anchor text. We truncate each anchor sentence to 80 characters and concatenate the resulting strings with spaces. Within each training fold, we fit a TF-IDF vocabulary of at most 8,000 unigrams using `TfidfVectorizer`. Term frequency is transformed as $1 + \log(\mathrm{tf})$ before inverse-document-frequency weighting. Both vocabulary construction and weighting are fitted using only the training fold.

Classifier settings. Concatenating the two components yields at most 8,117 input features for XGBoost. We use 400 trees, maximum depth 6, learning rate 0.05, row subsampling 0.8, and column subsampling 0.8, with a log-loss objective. Predicted probabilities are used directly as correctness scores. The model-identification variant uses the same features and hyperparameters with the multiclass `multi:softprob` objective.

Depth-specific comparison. For Op-XGB-Early, we evaluate depths $d \in \{10, 25, 50, 75, 100\}\%$. At each depth, we retain the first $\max(1, \lceil Md/100 \rceil)$ of the M spans in the trace, including structural spans. We then remove unlabeled spans from the operator sequence, recompute both feature components, and fit a new classifier under the same five-fold problem-level protocol. The text vocabulary is refitted for that depth and training fold. This provides a depth-specific comparison with the fixed OST; it is not a formal upper bound on prediction from operator features.

F Operator Usage by Correctness per Benchmark

Figure 8 shows operator frequency separately for correct and incorrect traces. INFERRING is consistently higher in correct traces; HYPOTHESIZING and BACKTRACKING are higher in incorrect traces. The gap is largest on commonsense benchmarks (ARC-C, MMLU-Pro: ∼24% incorrect vs. ∼14% correct for HYPOTHESIZING). On LiveCodeBench, CONSTRAINING is 10 pp higher in correct traces.

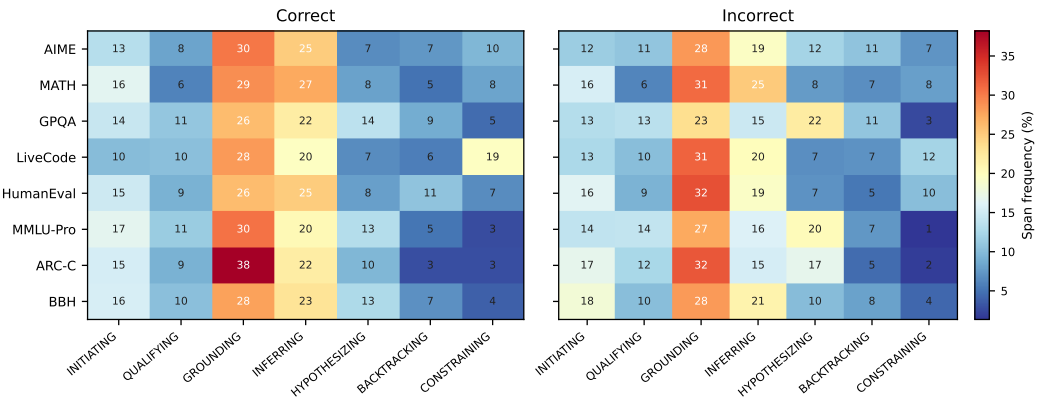


Figure 8: Operator frequency (%) per benchmark, split by correctness. Left: correct traces. Right: incorrect traces. Both panels share a color scale; rows are benchmarks and columns are operators.

G TEMPORAL OPERATOR PROFILES

Figure 9 shows operator density as a function of normalized trace position. GROUNDING dominates the early trace; INFERRING concentrates near the end. BACKTRACKING shows a midtrace spike in incorrect traces, consistent with exploration that is not followed by a correct answer.

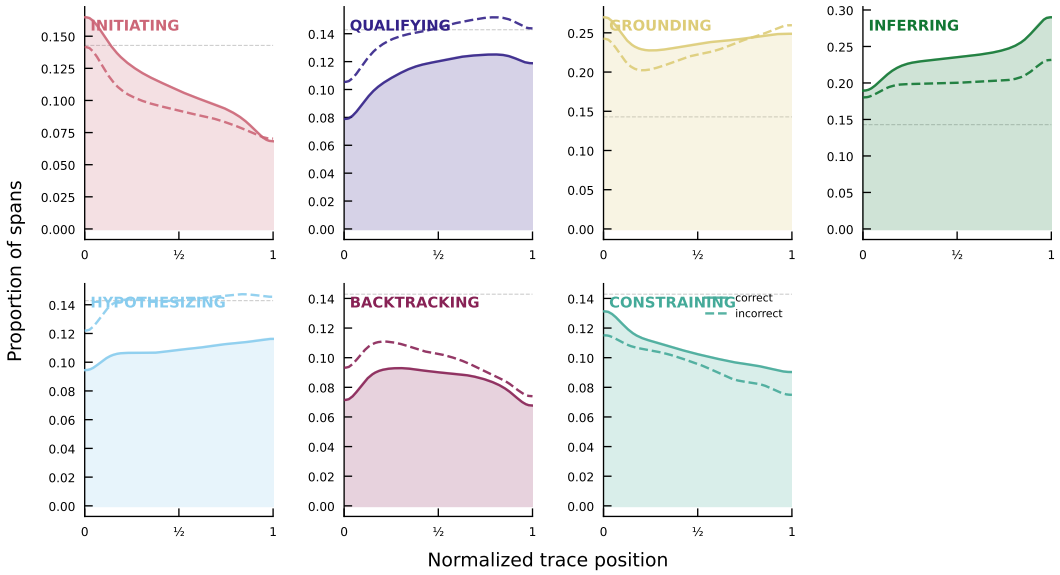


Figure 9: Operator density over normalized trace position, estimated with Gaussian KDE. Solid blue curves represent correct traces and dashed orange curves represent incorrect traces. The dotted reference indicates uniform usage (1/7).

H BACKTRACKING ANALYSIS DETAILS

We classify each BACKTRACKING and HYPOTHESIZING event on GPQA as LOCAL (re-checks a single calculation; method unchanged), SUB-PROBLEM (re-opens a specific case or branch), or GLOBAL (abandons the current solution strategy), using Claude Sonnet 4.6 as a judge.

Sample. Events are drawn from GPQA traces with stratification over (model, correctness), yielding n=1,294 BACKTRACKING and n=192 HYPOTHESIZING events.

Prompt. The judge receives the GPQA problem and the full reasoning trace. We mark a contiguous region beginning at the target span and ending before the next span of the same operator class, or at the end of the trace. The judge returns a JSON object containing a scope label and a rationale, using the system rubric reproduced below.

Operator	Local	Sub-Problem	Global	n
BACKTRACKING	85.4%	12.8%	1.6%	1,294
HYPOTHESIZING	80.2%	18.2%	1.6%	192

Table 7: Scope classification for BACKTRACKING and HYPOTHESIZING events on GPQA.

Prompt: Backtracking analysis judge prompt

You are analyzing reasoning traces from large language models solving GPQA (graduate-level science) questions. Your task is to classify the SCOPE of a single REVISION-ADJACENT region marked in the trace—a contiguous section where the model re-checks, considers an alternative, hypothesizes, qualifies, or hesitates.
The marked region (between `>>>MARKER<<<` and `<<<END>>>` markers) starts at the target operator event and ends just before the next event of the SAME operator type (or at the end of the trace). Read the surrounding context to decide what the model is actually re-thinking inside the marked region.

CLASSIFY THE SCOPE:

LOCAL: The model is re-checking a single calculation, fact lookup, or specific claim (e.g., "wait, $3 \times 7 = 21$ not 18"; "actually that should be oxygen"; "let me recompute that integral"; "I think this is the methyl group"). The OVERALL APPROACH is unchanged—same method, same plan, just verifying or fixing one step. Includes self-confirmation moves and tentative single-fact hypotheses.

SUB_PROBLEM: The model is re-opening or alternating among specific cases, branches, or sub-questions within the larger problem (e.g., "let me reconsider case 2"; "what about the limit where $x \to 0$?"; "could this be the (R) instead of (S) configuration?"). Strategy preserved but a discrete piece is redone or swapped.

GLOBAL: The model abandons or proposes to replace its current solution strategy/method with a fundamentally different approach (e.g., "this Lagrangian approach isn't working, let me try energy conservation"; "I shouldn't use the ideal-gas assumption—let me redo with van der Waals"; "I had the wrong mechanism entirely"; "let me use Gauss's law instead of Coulomb"). The high-level plan changes.

Output format: a single JSON object on one line:

```
{"scope": "LOCAL"|"SUB_PROBLEM"|"GLOBAL", "rationale": "<one short sentence>"}
```

Be decisive. Compare what the model is doing in the spans IMMEDIATELY before vs. after the marked region—that local context tells you what is actually changing. Judge each case on its merits without a default fallback category.

Prompt 1: System prompt used to classify the scope of BACKTRACKING and HYPOTHESIZING spans in GPQA reasoning traces in backtracking analysis.

I PER-DATASET BASELINE COMPARISON

Figure 10 shows WP-AUC (cross-dataset protocol) across all six baselines for each of the 6 benchmarks. BIG-Bench Hard and HumanEval are excluded (see §6.2 and Appendix B). Op-XGB and OST dominate on math-heavy and code benchmarks; the LLM self-judge is near chance on all datasets.

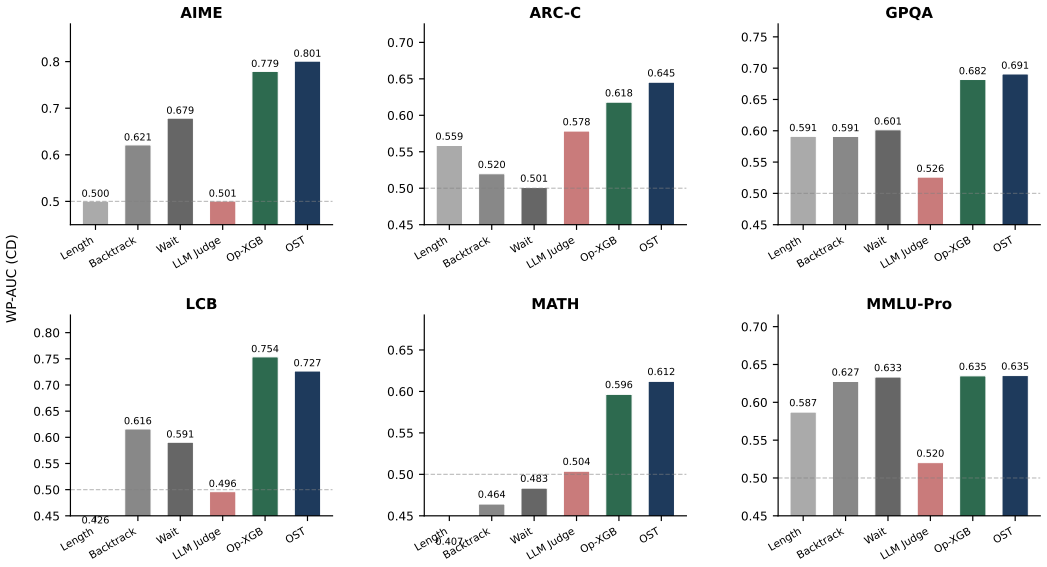


Figure 10: Per-benchmark WP-AUC (cross-dataset CV) for all six methods. Gray bars show the three scalar baselines and the LLM self-judge. Orange bars show Op-XGB, and blue bars show OST.

J SEGMENTATION WALKTHROUGH AND COMPLETE TRACE

Preprocessing. Normalize newlines, replace code and short inline math with `[CODE]` and `[MATH]`, and split at paragraph/line breaks and sentence-terminal punctuation. Lexical anchors begin with an uppercase character. Pivots use up to three lowercase alphabetic tokens, retaining apostrophes.

Boundaries and labels. A retained pivot opens a labeled span. Trace openings, list items, and questions following non-questions also open spans; lexical matches take precedence. Structural-only spans receive -1 and are excluded from prediction. Each span continues until the next boundary. The leading pivot alone determines its label, even if later clauses change function.

Example. Table 8 contains every stored span from a correctly answered R1-Distill-LLaMA-70B MATH problem about 35 students. Induced labels are preserved, including the counterintuitive assignment of "Therefore, the possible values..." to HYPOTHESIZING.

Table 8: Complete stored span text for a correctly answered MATH trace. Anchors are upright; continuation sentences are italicized.

Operator	Span text
UNLABELED	First, I recognize that there are 35 students in total at the math club meeting.
INITIATING	Let's denote the number of girls as G and the number of boys as B.

Operator	Span text (continued)
INFERRING	So, the equation is G + B = 35. *Next, it's given that the number of girls is a multiple of 13. This means G can be 0, 13, 26, or 39.*
QUALIFYING	However, since the total number of students is 35, G can't be 39.
HYPOTHESIZING	Therefore, the possible values for G are 0, 13, and 26. *The problem also states that there are more girls than boys. Since G > B and G + B = 35, we can substitute B with 35 - G.*
CONSTRAINING	This gives us G > 35 - G. *Simplifying this inequality, we find that G must be greater than 17.5.*
GROUNDING	Since the number of girls must be an integer, G must be at least 18. *Among the possible values of G (0, 13, 26), only 26 is greater than 17.5. Therefore, G = 26. Substituting this back into the total number of students, we get B = 35 - 26 = 9.*
INFERRING	Thus, there are 9 boys at the meeting.

Operator	BBH	MMLU-Pro	HumanEval	LiveCodeBench
Initiating	Let me calculate the distance between these two points.	Let me check if 442.02 comes from a different approach.	Let me check for 3 again. 3 in binary is 11 (two ones).	Let me consider the spf (smallest prime factor) method.
Qualifying	But the instruction "Always face forward" is confusing.	But the answer options are all absolute dollar amounts.	But if all are same, then the min and max are the same.	But the condition is A[j] - A[i] < M for the endpoints.
Grounding	I need to figure out their order based on their prices.	I need to find the time from discount date to maturity.	I need to check if there are any letters in the string.	I need to compute for mask with all bits set, who wins.
Inferring	So (B) is wrong because size is after origin and color.	So (C) and (G) are more likely to result in not guilty.	So 1e-9 is a bigger tolerance than the machine epsilon?	So "10101" is 21 in decimal, which is not a power of 5.
Hypothesizing	Alternatively, maybe I mixed up the order of the swaps.	Alternatively, I might be mixing up with another movie.	Alternatively, I can compute the factorial iteratively.	Alternatively, I might have an error in the recurrence.
Backtracking	But wait, I think country of origin comes before color.	But wait, another angle: the Rule Against Perpetuities.	But wait, I need to be careful about preserving spaces.	But wait, I shouldn't have a trailing space at the end.
Constraining	So we need three positions in increasing order after 2.	So we need the annuity factor for 20% to be at least 3.	So we need to count letters only, not other characters.	Now we need to also compute final position after moves.

Additional operator examples from the four remaining benchmarks. Together with Figure 1, this inventory contains one example of every operator from every dataset.

K EVALUATION

K.1 STABILITY ANALYSIS

Naming protocol. We test whether a cluster receives a consistent description when the example spans used to name it change. The seven cluster assignments remain fixed. For each cluster, we run Claude Sonnet 4.6 (`claude-sonnet-4-6`) at temperature zero under 30 exemplar seeds. Each prompt contains the same twelve highest-ranked anchor n-grams and five sampled spans. Only the five spans vary across seeds, so the experiment measures naming stability rather than clustering stability.

This procedure produces $30 \times 7 = 210$ descriptions. We embed each description with `text-embedding-3-small`, giving a 1,536-dimensional vector, and normalize it to unit length.

Cosine similarity is then the inner product of two vectors. The within-cluster comparison contains $\binom{30}{2} = 435$ pairs per operator, or 3,045 pairs in total. As a reference, we sample 2,000 pairs whose descriptions come from different operators.

A second control tests how much consistency arises from the naming prompt alone. We form seven disjoint random groups of 2,000 spans from the full type-A span pool and apply the same naming procedure. These groups mix the real operators in corpus-rate proportions. Their frequent openings therefore tend to be common phrases such as "now we need" and "let me think," rather than cues specific to one operator.

Statistical comparison. We compare the within-cluster similarities with each reference distribution using a one-sided Mann–Whitney U test. We also report the rank-biserial effect size,

$$r_{\mathrm{rb}} = \frac{2U}{n_1 n_2} - 1, \tag{9}$$

where n_1 and n_2 are the numbers of similarities in the two groups. This statistic ranges from -1 to $+1$ and, in the absence of ties, equals $2P(A > B) - 1$ for draws A and B from the two distributions.

The pairwise similarities are not independent: the 435 pairs within a cluster share only 30 underlying embeddings. Consequently, the U-test p-values are anticonservative. We retain them for completeness but emphasize the effect sizes as descriptive measures of separation; their calculation does not require independent pairs.

Results. Descriptions of the same operator are more similar than descriptions of different operators or random groups (Figure 11). Median similarity is 0.660 within clusters, 0.467 across clusters, and 0.565 for the random-group control. The corresponding within-minus-reference effect sizes are $r_{\mathrm{rb}} = +0.72$ and $+0.39$, with reported one-sided $p \approx 0$ and $p < 10^{-149}$. Approximately 86% of within-cluster pairs are more similar than an across-cluster pair, and 69% are more similar than a pair from the random-group control.

All seven operators exceed the across-cluster baseline. Their effect sizes range from $+0.22$ for GROUNDING to $+1.00$ for BACKTRACKING, with reported per-cluster $p < 10^{-12}$. Random groups also yield more similar descriptions than the across-cluster reference, with an effect size of $+0.44$ and a median gap of 0.098. Thus the prompt produces some consistency even for mixed groups, but the learned clusters add further semantic consistency.

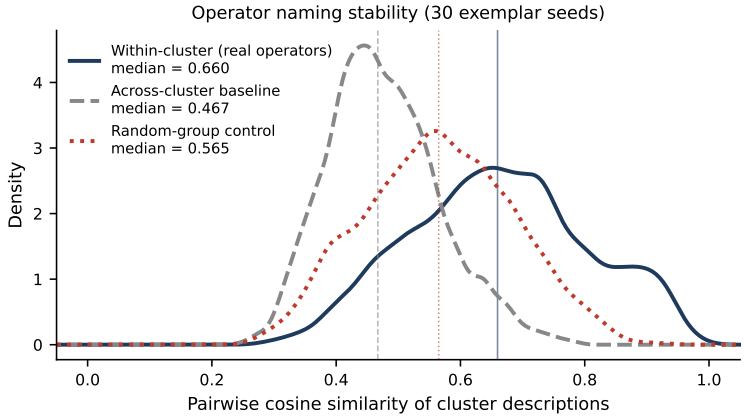


Figure 11: Naming stability: pairwise cosine similarity distributions between cluster-description embeddings generated under 30 exemplar seeds (`text-embedding-3-small`, L_2-normalized). The *within-cluster* distribution (median $= 0.660$) is well separated from both the *across-cluster* baseline (median $= 0.467$) and the *random-group control* (median $= 0.565$), which applies the same naming protocol to seven disjoint random span groupings. The comparison tests naming consistency with the cluster assignments held fixed.

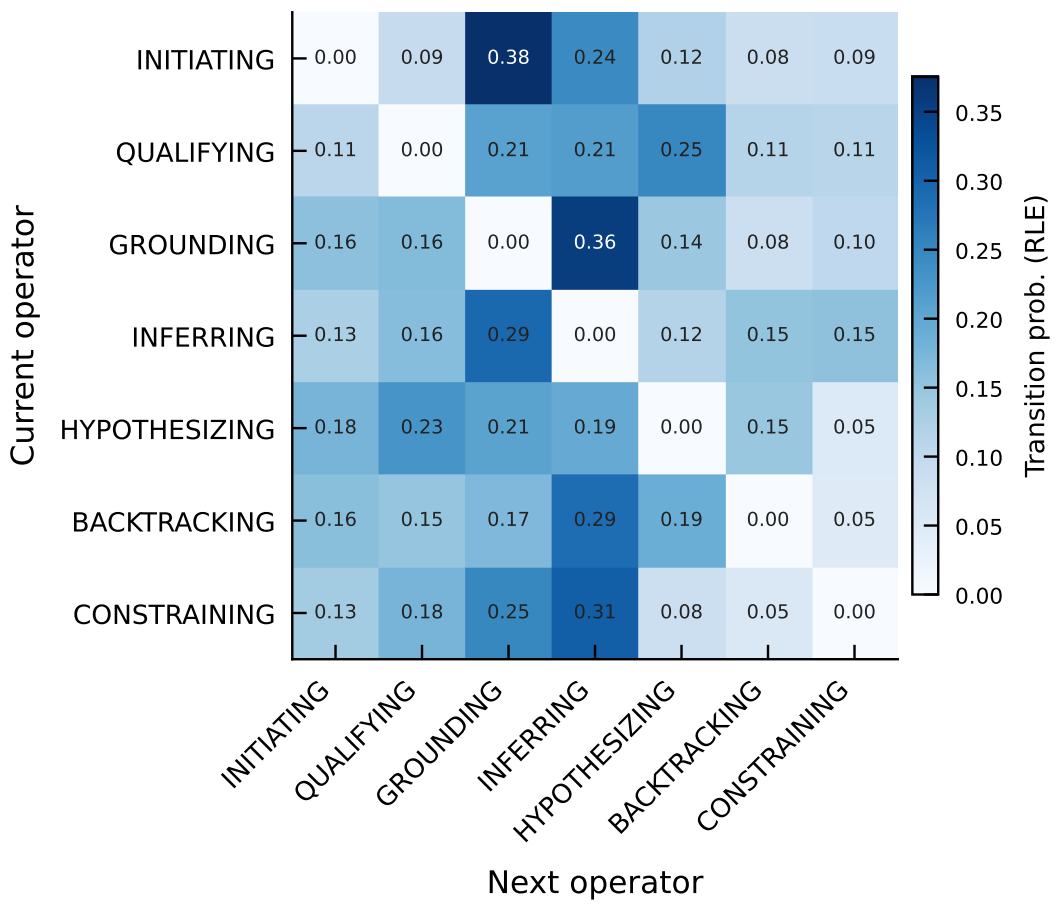


Figure 12: Operator transition probabilities after consecutive repeats are collapsed. Rows denote the current operator and columns the next operator. Entry (i, j) is the probability that the next run has operator j, given that the current run has operator i.

L Additional Analyses

L.1 Transition Structure and Run-Length Statistics

Figure 12 shows transition probabilities between operator runs. The associated run-length summaries show that GROUNDING has the longest self-runs, consistent with sustained fact-anchoring across consecutive spans. In contrast, BACKTRACKING is nearly always a single span. This pattern is consistent with a brief revision signal rather than a sustained change in strategy. Transitions from HYPOTHESIZING to INFERRING are more frequent than most other off-diagonal transitions, consistent with tentative scenarios being followed by conclusions.

L.2 Model Identification Confusion Matrix

Figure 13 shows the confusion matrix for the 12-way model identification task under problem-level 5-fold CV using the full Op-XGB feature recipe (Appendix E). The classifier reaches accuracy $= 79.9\%$ and macro-AUC $= 0.987$. Errors concentrate within model families. The R1-Distill-Llama-70B / R1-Distill-Qwen-32B pair contributes the largest off-diagonal mass: R1-Distill-Llama-70B traces are predicted as R1-Distill-Qwen-32B at 0.41 and the reverse at 0.29, consistent with their shared distillation lineage from DeepSeek-R1. The Claude-Haiku-4.5 / Claude-Sonnet-4.5 pair shows mutual confusion in the 0.21–0.26 range, and Kimi-K2.5 is predicted as Kimi-K2 at 0.10 (with the reverse at 0.05). Off-diagonal mass outside these family clusters is small. Per-model

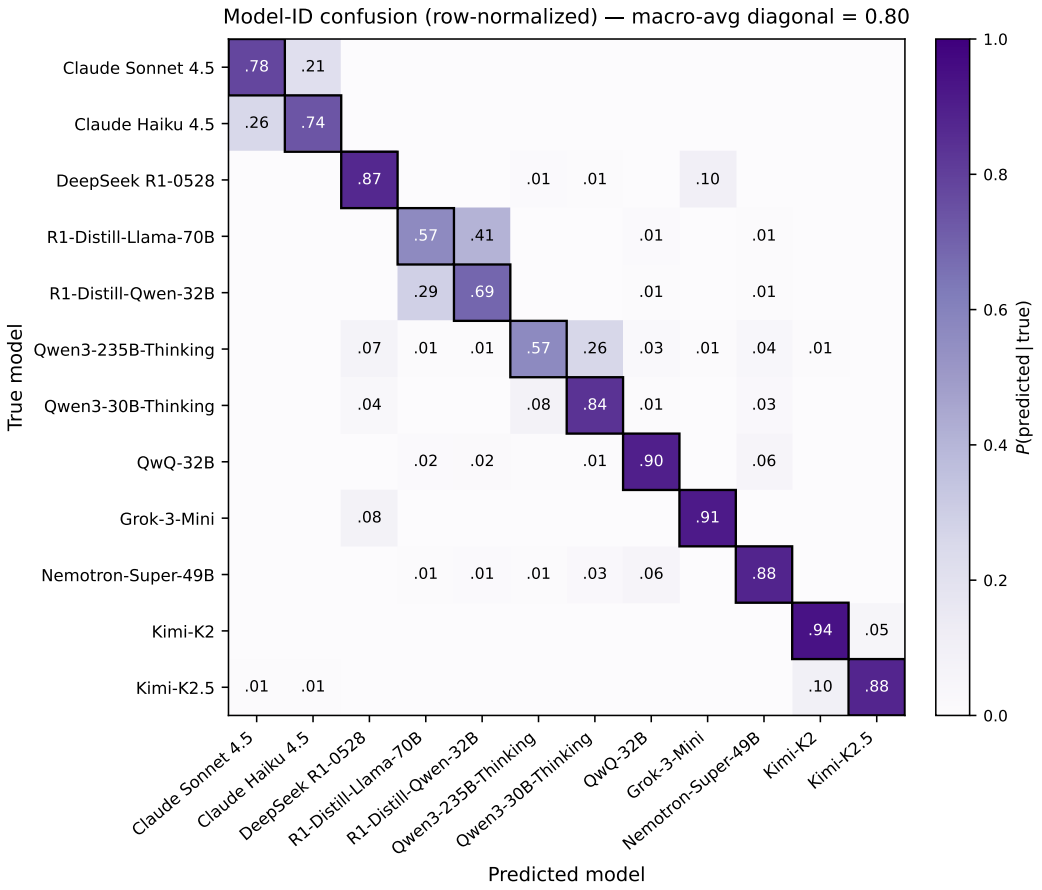


Figure 13: Model identification confusion matrix (row-normalized) under problem-level 5-fold CV with the full Op-XGB feature recipe (Appendix E). Rows denote the true model and columns the predicted model. Accuracy = 0.799, macro-AUC = 0.987. Errors concentrate within model families (R1-Distill pair, Claude pair, Kimi pair).

AUCs span 0.967 (R1-Distill-Llama-70B) to 0.997 (Kimi-K2, Kimi-K2.5, Grok-3-Mini, all tied at this rounding).

M Correctness-Prediction Protocol and Detailed Comparisons

The correctness-prediction subset contains 33,209 traces across six benchmarks. We assign all attempts at a problem to the same fold. Cross-dataset training pools benchmarks within each fold; in-dataset training fits one predictor per benchmark. WP-AUC averages AUC across problems with both correct and incorrect responses, as defined in Equation 10.

Within-problem AUC. Let $\mathcal{Q}_{\pm}$ contain problems with both correct and incorrect attempts, and let AUC_q be the AUC of predictor scores against correctness labels within problem q. We give each eligible problem equal weight:

$$\text{WP-AUC} = \frac{1}{|\mathcal{Q}_{\pm}|} \sum_{q \in \mathcal{Q}_{\pm}} \mathrm{AUC}_q \,. \tag{10}$$

Problems whose attempts are all correct or all incorrect have no within-problem ranking comparison and are excluded from this average. Thus the metric measures discrimination among attempts at the same problem, rather than between easy and hard problems.

Baselines. The length baseline uses the logarithm of the response character count. The backtracking baseline uses the fraction of spans labeled BACKTRACKING, and the wait baseline counts "wait"-prefixed spans. Each scalar feature is passed to logistic regression under the same five-fold problem-level protocol. SelfCheck (Miao et al., 2024) instead asks the generating model to assess its response from the problem and reasoning text, without the gold answer; it requires no fitting. Op-XGB and OST use the representations defined in Section 6.

Detailed early-prediction comparison. Op-XGB-Early is refitted at each depth, whereas the OST weights remain fixed after training on full sequences. The Op-XGB-Early WP-AUCs at 10%, 25%, 50%, 75%, and 100% depth are, respectively, 0.620, 0.647, 0.661, 0.682, and 0.699. The reported OST-minus-Op-XGB-Early differences at those depths are -0.015, -0.016, $+0.003$, $+0.006$, and $+0.003$. Thus the depth-specific tree models have higher point estimates at the two shortest prefixes, while the sequence model matches or slightly exceeds them from 50% depth onward. This comparison is notable because the tree models also receive anchor text and are optimized separately for each depth.

Scalar scores across protocols. A monotone transformation of one set of scores does not change its AUC. However, CD and ID evaluations fit different models and may aggregate predictions from different folds, so this invariance does not imply equal reported scores across protocols. Table 3 preserves the measured CD and ID values.

N HUMAN VALIDATION AND ANNOTATION LIMITS

Protocol. Two annotators applied the operator codebook to held-out spans with induced labels withheld. Annotator A1 labeled 250 spans and A2 labeled 251, with 250 spans in common. We compare both on this shared subset. The full validation pool contains 350 spans, balanced across the seven induced operators.

A2 marked two shared spans as `NONE`. We retain these responses as disagreements with the induced operator and with A1; unannotated spans are excluded. Agreement with the induced labels measures label recovery, not agreement with independent ground truth. Inter-annotator agreement is computed separately, without adjudication.

Agreement. On the shared set, A1 agrees with 156/250 induced labels (62.4%; 95% Wilson interval 56.3–68.2), and A2 agrees with 130/250 (52.0%; 45.8–58.1). Their Cohen's kappas are 0.562 and 0.443, with 95% bootstrap intervals $[0.490, 0.630]$ and $[0.371, 0.512]$. The annotators agree with each other on 145/250 spans (58.0%; 51.8–64.0), with $\kappa = 0.504$ ($[0.433, 0.574]$). Bootstrap intervals use 20,000 paired span resamples with seed 20260924.

Table 9: Human label recovery on the same 250 spans. Counts refer to induced operators; precision, recall, and F1 treat these labels as the reference. A1 and A2 denote the two annotators.

		A1			A2		
Operator	n	Prec. (%)	Rec. (%)	F1	Prec. (%)	Rec. (%)	F1
INITIATING	35	100.0	77.1	0.87	80.0	68.6	0.74
QUALIFYING	37	61.0	67.6	0.64	62.5	40.5	0.49
GROUNDING	33	44.2	57.6	0.50	36.2	63.6	0.46
INFERRING	36	45.8	61.1	0.52	42.6	63.9	0.51
HYPOTHESIZING	35	51.9	40.0	0.45	52.9	25.7	0.35
BACKTRACKING	32	81.6	96.9	0.89	71.1	100.0	0.83
CONSTRAINING	42	69.2	42.9	0.53	30.0	14.3	0.19

Operator differences. Initiating and backtracking have the highest F1 for both annotators (Table 9). Agreement is lower for hypothesizing and constraining: their F1 scores are 0.45/0.35 and 0.53/0.19 for A1/A2, respectively. The expanded audit therefore supports uneven semantic recovery across the vocabulary rather than uniform agreement on all seven operators.

Additional response and abstentions. Including A2's additional response gives 130/251 agreement with induced labels (51.8%; $\kappa = 0.440$). Excluding the two NONE responses gives 130/249 (52.2%; $\kappa = 0.445$); inter-annotator agreement on the remaining shared spans is 145/248 (58.5%; $\kappa = 0.509$). Thus, these responses have little effect on the aggregate results.

O VOCABULARY COVERAGE

The first 6,000 stored traces contain 3,732 Claude-Haiku and 2,268 Claude-Sonnet responses. This convenience sample is not representative of the full corpus. Boundaries occur at 36% of sentence-like units; most sentences continue an earlier span.

Structural-only boundaries comprise 28.0% of spans. Of these, 97.1% are list items, 2.1% self-posed questions, and 0.7% trace openings (rounded). Math markup, formula placeholders, or equals signs occur in 37.6% of list-item anchors.

Table 10: Span-label proportions in the ordered 6,000-trace coverage sample. Structural-only spans are not a learned eighth operator.

Label	Share (%)
GROUNDING	39.4
Unlabeled structural boundary	28.0
INITIATING	15.2
INFERRING	7.9
HYPOTHESIZING	3.5
QUALIFYING	2.7
CONSTRAINING	2.4
BACKTRACKING	0.8

Computation, fact retrieval, and example testing are distributed across labels according to their opening connective. For example, "Let's search memory" is initiating, while "I recall that for lithium enolates of cyclohexanone" is hypothesizing. The dataset filter can exclude domain-specific moves such as complexity analysis. Self-questioning remains unlabeled unless its opening matches a retained pivot.

P POST-TRAINING COMPARISON WITH OLMO

Checkpoints and evaluation set. The General checkpoint is `allenai/Olmo-3-7B-RL-Zero-General`. The Math checkpoint is `allenai/Olmo-3.1-7B-RL-Zero-Math`. The Code checkpoint is `allenai/Olmo-3.1-7B-RL-Zero-Code`. Their model cards identify a shared OLMo-3-7B base (Allen Institute for AI, 2025). General was reinforced on open-ended chat prompts scored by an LM judge, whereas Math and Code used domain-specific prompts with rule-based and execution-based verification, respectively. We analyze 254 traces across AIME and LiveCodeBench: 44 General, 40 Math, and 57 Code traces on AIME, and 48 General, 22 Math, and 43 Code traces on LiveCodeBench. The generation records identify checkpoints by repository name but do not preserve immutable Hugging Face revision hashes.

Trace generation. We served each checkpoint locally in BF16 with vLLM's OpenAI-compatible server and the Transformers OLMo 3 implementation. Each problem was supplied as a single user message through the checkpoint's original chat template. Generation was deterministic (temperature 0, top-$p = 1$, one completion per prompt). We used a staged procedure to obtain complete traces: an initial generation allowed up to 8,192 new tokens in a 32,768-token context; when more generation was needed, we generated afresh from the original prompt with a 16,384-token allowance; remaining continuations used the saved 16K text as a re-tokenized prefix in a 65,536-token context, reserving 64 tokens for the context boundary.

Operator assignment and aggregation. We normalize Markdown list prefixes before segmentation and apply the frozen $K = 7$ three-token pivot lookup to every trace. We do not re-run

pivot discovery, embedding, clustering, or operator naming. Each table entry is the mean per-trace fraction of an operator among the seven labeled operators; structural spans are excluded from the denominator.

Table 11: Full operator distributions for the OLMo post-training analysis. Values are macro-averaged percentages across the n analyzed traces in each cell; structural spans are excluded.

Dataset	Variant	n	Init	Qual	Ground	Infer	Hypo	Back	Constr
AIME	General	44	8.4	8.6	48.2	15.3	11.3	1.7	6.6
	Math	40	8.1	13.0	30.7	22.6	5.2	17.8	2.6
	Code	57	11.6	9.7	27.9	22.2	7.4	16.8	4.5
LiveCodeBench	General	48	3.3	8.7	57.6	6.4	15.3	1.2	7.5
	Math	22	5.6	11.7	23.9	34.4	4.6	18.5	1.3
	Code	43	4.1	4.1	33.2	31.1	5.3	21.7	0.6

Accuracy among these traces is 22.7%, 70.0%, and 45.6% for General, Math, and Code on AIME, and 10.4%, 59.1%, and 25.6% on LiveCodeBench. We report results in Table 11.

Q Hyperparameter Sensitivity

We vary the operator count ($K = 4$–12), pivot length ($n = 1$–5), minimum trace frequency (25–400), minimum dataset count (1–5), and vocabulary size (1,000–4,000). We also compare K-means with agglomerative clustering and HDBSCAN, and compare `e5-small-v2` with `all-MiniLM-L6-v2` and `bge-small-en-v1.5`. Finally, 24 sampled joint configurations vary all four pivot/filter parameters together.

Evaluation. We use problem-level 5-fold cross-validation for model identification and correctness, with operator features only (117 at $K = 7$). Excluding anchor TF-IDF isolates the representation; these scores are not directly comparable to full Op-XGB's 0.987 model-ID AUC.

XGBoost uses 400 trees, depth 6, learning rate 0.05, and row/column subsampling 0.8. Tables report mean and standard deviation over three initialization seeds where applicable; deterministic methods have one result. Correctness excludes BIG-Bench Hard and HumanEval.

Selection and tradeoffs. $K = 7$ maximizes judge agreement over the original $K = 6$–11 sweep, but does not maximize every downstream metric: larger K improves model identification, and operator-only correctness is slightly higher around $K = 8, 9$. Retraining the OST gives WP-AUC 0.6928, 0.7036, and 0.6942 at $K = 6, 7, 8$, respectively; agglomerative clustering at $K = 7$ yields 0.6802. Seven operators favor interpretability; the sweep does not establish global optimality or significant differences between nearby settings.

Two- and three-token pivots give similar model-ID AUC (0.9477 and 0.9472); five-token pivots fall to 0.8924. Three tokens outperform two on correctness in this sweep, although the single-token variant is slightly higher still.

Varying filter thresholds changes model-ID AUC only modestly. Joint configurations span 0.909–0.950 in model-ID AUC and 0.618–0.658 in correctness (medians 0.935 and 0.642).

Alternative encoders yield different partitions (ARI 0.350–0.418 versus the default) with similar downstream scores. HDBSCAN is a counterexample to uniform robustness: this configuration produces three remapped labels and reduces model-ID AUC to 0.8270 and correctness to 0.5723; its noise label is treated as a separate label by this implementation.

Table 12: Number of operators. Operator-only feature evaluation; mean $\pm$ SD over initialization seeds where applicable.

Setting	Pivots	Model-ID AUC	WP-AUC
4	5464	0.8952 ± 0.0077	0.6259 ± 0.0055
5	5464	0.9215 ± 0.0001	0.6504 ± 0.0026

Setting	Pivots	Model-ID AUC	WP-AUC
6	5464	0.9373 ± 0.0000	0.6487 ± 0.0023
7	5464	0.9473 ± 0.0006	0.6479 ± 0.0032
8	5464	0.9502 ± 0.0004	0.6604 ± 0.0003
9	5464	0.9526 ± 0.0006	0.6610 ± 0.0019
10	5464	0.9552 ± 0.0003	0.6537 ± 0.0025
11	5464	0.9553 ± 0.0001	0.6516 ± 0.0012
12	5464	0.9572 ± 0.0012	0.6562 ± 0.0035

Table 13: Pivot length. Operator-only feature evaluation; mean $\pm$ SD over initialization seeds where applicable.

Setting	Pivots	Model-ID AUC	WP-AUC
1	652	0.9373 ± 0.0009	0.6521 ± 0.0049
2	4159	0.9477 ± 0.0003	0.6440 ± 0.0021
3	5464	0.9472 ± 0.0007	0.6499 ± 0.0015
4	3770	0.9138 ± 0.0006	0.6277 ± 0.0017
5	2382	0.8924 ± 0.0017	0.6185 ± 0.0038

Table 14: Minimum trace frequency. Operator-only feature evaluation; mean $\pm$ SD over initialization seeds where applicable.

Setting	Pivots	Model-ID AUC	WP-AUC
25	19332	0.9438 ± 0.0003	0.6590 ± 0.0040
50	10604	0.9444 ± 0.0017	0.6519 ± 0.0042
100	5464	0.9472 ± 0.0007	0.6499 ± 0.0015
200	2698	0.9425 ± 0.0004	0.6514 ± 0.0021
400	1277	0.9385 ± 0.0002	0.6446 ± 0.0006

Table 15: Minimum dataset count. Operator-only feature evaluation; mean $\pm$ SD over initialization seeds where applicable.

Setting	Pivots	Model-ID AUC	WP-AUC
1	5908	0.9441 ± 0.0023	0.6526 ± 0.0026
2	5771	0.9470 ± 0.0032	0.6513 ± 0.0002
3	5464	0.9472 ± 0.0007	0.6499 ± 0.0015
4	5170	0.9418 ± 0.0019	0.6474 ± 0.0030
5	4634	0.9393 ± 0.0004	0.6525 ± 0.0052

Table 16: Vocabulary size. Operator-only feature evaluation; mean $\pm$ SD over initialization seeds where applicable.

Setting	Pivots	Model-ID AUC	WP-AUC
1000	5115	0.9338 ± 0.0027	0.6534 ± 0.0012
1500	5337	0.9392 ± 0.0026	0.6471 ± 0.0034
2000	5464	0.9472 ± 0.0007	0.6499 ± 0.0015
3000	5570	0.9464 ± 0.0024	0.6494 ± 0.0030
4000	5600	0.9494 ± 0.0021	0.6514 ± 0.0019

Table 17: Clustering algorithm. Operator-only feature evaluation; mean ± SD over initialization seeds where applicable.

Setting	Labels	ARI	Model-ID AUC	WP-AUC
agglomerative	7	0.191	0.9205	0.6293
hdbscan	3	0.053	0.8270	0.5723
kmeans	7	0.971	0.9473 ± 0.0006	0.6479 ± 0.0032

Table 18: Sentence encoder. Operator-only feature evaluation; mean ± SD over initialization seeds where applicable.

Setting	Labels	ARI	Model-ID AUC	WP-AUC
bge	7	0.350	0.9385 ± 0.0003	0.6356 ± 0.0020
e5	7	0.971	0.9473 ± 0.0006	0.6479 ± 0.0032
minilm	7	0.418	0.9408 ± 0.0001	0.6467 ± 0.0015

Table 19: Joint configurations. Operator-only feature evaluation; mean ± SD over initialization seeds where applicable. Here n is pivot length, f minimum trace frequency, d minimum datasets, and v vocabulary size.

Setting	Pivots	Model-ID AUC	WP-AUC
n=2,f=100,d=3,v=2000	4159	0.9475	0.6455
n=2,f=200,d=5,v=1000	2187	0.9421	0.6357
n=2,f=25,d=1,v=1500	11738	0.9345	0.6383
n=2,f=25,d=2,v=1000	9846	0.9363	0.6550
n=2,f=25,d=3,v=4000	10842	0.9466	0.6369
n=2,f=400,d=1,v=2000	1427	0.9454	0.6459
n=2,f=400,d=2,v=4000	1438	0.9490	0.6459
n=2,f=50,d=1,v=1500	7141	0.9499	0.6505
n=3,f=100,d=2,v=1000	5385	0.9307	0.6576
n=3,f=100,d=2,v=2000	5771	0.9497	0.6511
n=3,f=200,d=1,v=1500	2773	0.9387	0.6503
n=3,f=200,d=2,v=2000	2795	0.9430	0.6477
n=3,f=200,d=5,v=1500	2457	0.9346	0.6530
n=3,f=400,d=1,v=1000	1277	0.9302	0.6512
n=3,f=400,d=2,v=3000	1319	0.9388	0.6432
n=3,f=400,d=3,v=1000	1237	0.9281	0.6416
n=4,f=100,d=1,v=4000	4297	0.9200	0.6322
n=4,f=200,d=2,v=1500	1805	0.9203	0.6193
n=4,f=200,d=5,v=1500	1511	0.9157	0.6176
n=4,f=25,d=1,v=1500	20258	0.9255	0.6307
n=4,f=25,d=4,v=2000	12825	0.9173	0.6308
n=4,f=400,d=2,v=1000	721	0.9114	0.6221
n=4,f=50,d=4,v=3000	7263	0.9168	0.6233
n=4,f=50,d=5,v=4000	5966	0.9093	0.6189

R System-Prompt Perturbations of Model Identification

We train the full 12-way Op-XGB classifier on the reference corpus and freeze it. Qwen3-30B-Thinking, Kimi-K2.5, and R1-0528 each generate new responses to 30 AIME and 40 GPQA problems under seven conditions, totaling $3 \times 70 \times 7 = 1{,}470$ traces. These are new generations, not a held-out-problem evaluation; the goal is perturbation robustness of the frozen classifier. Chance accuracy is 8.3%.

The conditions are: no system prompt; expert, high-stakes, and careful-reasoner personas; style-only evasion; evasion with disclosure that the classifier uses sentence openers; and mimicry with

disclosure plus twelve characteristic openers mined from a target model. Targets are cross-family: Qwen mimics Kimi, Kimi mimics R1, and R1 mimics Qwen. The target phrases are mined from the reference corpus, without changing the classifier after seeing test responses.

Prompt text.

> **Expert:** You are a world-class expert with deep mastery of this subject.
>
> **Stakes:** This is a high-stakes problem where accuracy is critical.
>
> **Careful:** You are an exceptionally careful and self-critical thinker.
>
> **Evade blind:** Your reasoning will be analyzed by a classifier that identifies which AI model produced it. Make your reasoning style as unrecognizable as possible: avoid your habitual phrasing and stylistic tics. Still solve the problem correctly.
>
> **Evade method:** Your reasoning will be analyzed by a classifier that identifies which AI model produced it from the first few words of each sentence (discourse markers such as 'Wait', 'So', 'Alternatively', 'Let me check'). Vary your sentence openers and avoid your habitual discourse markers. Still solve the problem correctly.
>
> **Mimic:** Your reasoning will be analyzed by a classifier that identifies which AI model produced it from the first few words of each sentence. Evade it by imitating a different model's reasoning style: where natural, begin your sentences with openers such as: *[target opener list]*. Avoid your own habitual openers. Still solve the problem correctly.

Table 20: Identification accuracy (%) under system prompts, 70 traces per cell. Parentheses give an operator-only classifier without anchor TF-IDF.

Condition	Qwen3-30B	Kimi-K2.5	R1-0528
No system prompt	91 (71)	86 (89)	94 (73)
Expert persona	93 (73)	90 (77)	91 (70)
High-stakes persona	99 (83)	87 (70)	94 (70)
Careful persona	94 (81)	93 (80)	96 (83)
Evade, style only	93 (74)	99 (79)	94 (74)
Evade, mechanism disclosed	83 (61)	99 (71)	91 (73)
Mimic target openers	60 (51)	94 (77)	94 (69)

Identification accuracy remains between 83% and 99% under persona and evasion prompts. Mimicry has a larger effect on Qwen3-30B, reducing identification to 60% and causing 13% of its traces to be attributed to Kimi. Kimi and R1 remain at 94%, with none of their traces attributed to the target model. Task accuracy stays within six percentage points of the no-prompt baseline in every condition.

The generated text suggests that prompt compliance contributes to these differences. Qwen uses essentially no target openers on AIME. Kimi, in contrast, increases its use of target openers from about 0.1 to roughly 100 per trace and remains identifiable. The operator-only comparison in Table 20 provides a separate measure of structural attribution, but does not prove invariance of every operator statistic.

An additional R1-Distill-LLaMA-70B endpoint produced a large attribution shift even without a system prompt and was excluded from this perturbation comparison. An attribution anomaly alone does not prove that a provider substituted its model. More generally, we do not test arbitrary fixed sentence prefixes, forced templates, or post-generation rewriting; all can directly disrupt the lexical boundary rule. The experiment therefore establishes robustness only over the tested prompt family.

S Operator-Based Selection at Inference Time

Generation pool and scorer. We generate 3,847 attempts from Qwen3-30B-Thinking, Kimi-K2.5, R1-0528, and R1-Distill-LLaMA-70B on 32 problems (16 AIME, 16 LiveCodeBench). Temperature is 0.7, the cap is 32,000 tokens per attempt, and total generation is 40.4M tokens. Excluding capped attempts leaves 872/907 AIME and 2,930/2,940 LiveCodeBench responses.

An XGBoost scorer uses the 117 operator features under leave-problem-out folds. Selection ranks attempts within model–problem cells; the main correctness analysis instead pools models within each problem.

Matched-attempt best-of-N. For each model–problem cell, we draw $N \in \{1, 2, 3, 5, 8, 12\}$ attempts with replacement in 2,000 simulations. Policies select the highest operator score, longest trace, or a random attempt. Paired comparisons use identical candidates and token spend; costs differ across N, and scorer computation is additional.

Intervals use 5,000 paired bootstrap samples over problems, after averaging models per problem. Marginal curves and paired differences are simulated separately, so rounding and Monte Carlo variation can produce small discrepancies.

Table 21: AIME: best-of-N selection and paired differences (95% bootstrap intervals). Marginal curves and paired differences use separate Monte Carlo estimates.

N	Random	Length	Operator	Δ vs. random	Δ vs. length
1	0.603	0.604	0.604	–	–
2	0.604	0.630	0.630	$+0.024$ $[+0.013, +0.035]$	-0.001 $[-0.021, +0.017]$
3	0.603	0.636	0.637	$+0.033$ $[+0.014, +0.052]$	$+0.001$ $[-0.029, +0.029]$
5	0.605	0.636	0.642	$+0.037$ $[+0.010, +0.064]$	$+0.005$ $[-0.035, +0.044]$
8	0.605	0.631	0.644	$+0.040$ $[+0.006, +0.073]$	$+0.013$ $[-0.037, +0.063]$
12	0.605	0.627	0.646	$+0.040$ $[+0.002, +0.078]$	$+0.018$ $[-0.040, +0.076]$

Table 22: LiveCodeBench: best-of-N selection and paired differences (95% bootstrap intervals). Marginal curves and paired differences use separate Monte Carlo estimates.

N	Random	Length	Operator	Δ vs. random	Δ vs. length
1	0.343	0.342	0.342	–	–
2	0.341	0.348	0.360	$+0.016$ $[+0.002, +0.031]$	$+0.013$ $[-0.009, +0.035]$
3	0.344	0.348	0.366	$+0.025$ $[-0.001, +0.051]$	$+0.020$ $[-0.016, +0.056]$
5	0.340	0.347	0.371	$+0.030$ $[-0.012, +0.073]$	$+0.024$ $[-0.031, +0.081]$
8	0.344	0.348	0.372	$+0.030$ $[-0.032, +0.090]$	$+0.028$ $[-0.046, +0.107]$
12	0.344	0.346	0.372	$+0.030$ $[-0.048, +0.108]$	$+0.028$ $[-0.060, +0.124]$

On AIME, gains over random selection are 2.4 points at $N = 2$ and 4.0 at $N = 12$, with intervals excluding zero. LiveCodeBench gains 1.6 points at $N = 2$ (95% CI 0.2–3.1); intervals at larger N include zero.

No comparison against trace length excludes zero. Increasing AIME N from 3 to 12 adds 1.0 point ($[-1.2, +3.0]$), providing no clear evidence of further gains.

Verifier diagnostics. We compare scorers within model–problem cells that contain both correct and incorrect uncapped attempts. On AIME, operator features achieve an AUC of 0.738, compared with 0.716 for trace length and 0.566 for the held-out OST. Shuffling operator order while retaining frequencies reduces the operator scorer's AUC to 0.647. The corresponding AUCs on LiveCodeBench are 0.568 for operator features, 0.521 for length, 0.544 for OST, and 0.543 for shuffled sequences.

Although the AIME point estimates favor operator features, the sample does not resolve an advantage over either control. The operator-minus-length difference is 0.023, with a 95% interval of $[-0.114, +0.159]$. The operator-minus-shuffled difference is 0.092, with an interval of $[-0.016, +0.196]$. Both include zero.

Voting. On AIME, voting among the top three operator-ranked attempts beats a random three by 3.3 points ($[0.4, 6.4]$); top-five voting gains 1.9 points ($[0.4, 3.8]$). All candidates must first be generated, so these results imply no token savings.

Score-weighted full voting leaves accuracy unchanged at 0.703. Selecting one attempt from the full pool reaches 0.656, below full self-consistency at 0.703.

Abort-and-restart. The OST checks every 2,000 reasoning characters and aborts after two scores below 0.50. All 32 test problems are excluded from training. A 160k-token budget per model–problem includes aborted tokens.

Accuracy is 0.391 versus 0.456 for a random completed attempt at matched spend: -0.065 ($[-0.135, +0.002]$). The tested restart policy does not improve accuracy; the positive best-of-N result uses the separate operator-feature scorer.

APPENDIX REFERENCES

Anthropic. Introducing Claude 4. https://www.anthropic.com/news/claude-4, May 2025. Blog post, May 22, 2025.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, et al. Llama-nemotron: Efficient reasoning models. *arXiv preprint arXiv:2505.00949*, 2025.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Kimi Team. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Qwen Team. QwQ-32B: Embracing the power of reinforcement learning. https://qwenlm.github.io/blog/qwq-32b/, mar 2025. Blog post, March 6, 2025.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

xAI. Grok 3 beta — the age of reasoning agents. https://x.ai/news/grok-3, February 2025. Blog post, February 19, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.