ReasonOps: Operator Segmentation for LLM Reasoning Traces

Anonymous Author(s)
Affiliation
Address
email

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 develop ReasonOps, an unsupervised, expressive method for annotating chain-of-thought traces, providing succinct universal operators. 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 reflective operators are more helpful on hard problems and harm performance on easy problems. Operator sequences are highly model-identifying: a classifier trained on operator distributions alone recovers 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 $= 0.701$ and 0.801 on AIME specifically. ReasonOps further enables early quality estimation well before the trace completes: we predict at WP-AUC $= 0.664$ for only 50% of the trace. 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 [1, 2, 3, 4]. 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 [5, 6, 3]. This comes at an increasing demand for monitoring and oversight of LLM decision-making processes [7, 8, 9].

Since the advent of chain-of-thought (CoT) prompting [5], 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

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

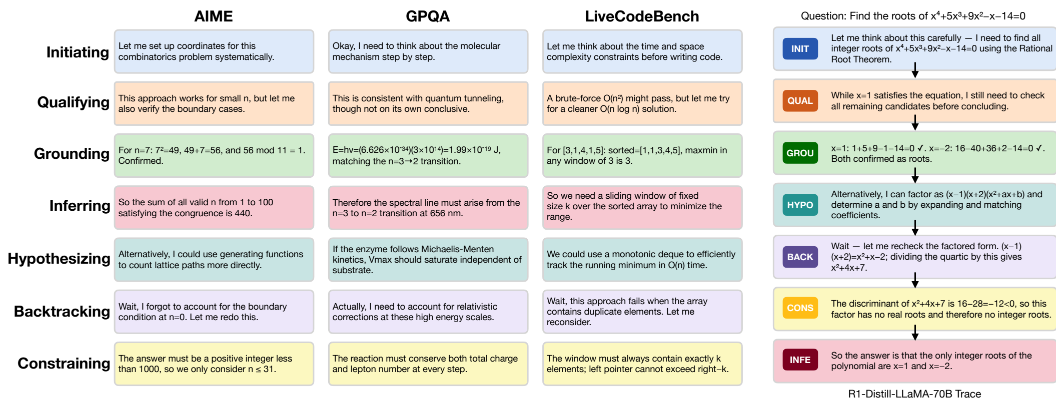

Figure 1: (Left) representative operator units extracted from AIME, GPQA, and LiveCodeBench. (Right) representative reasoning trace extracted from R1-Distill-LLaMA-70B on MATH dataset.

access to model weights. These traces have been shown to contain rich information [10, 11], 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 syntactic structures that are overfit to current models or domains [12, 13, 14, 15, 16]. 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 [17]. 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%). Operator sequences identify the source model with macro-AUC $= 0.987$, revealing model-family reasoning fingerprints. We show that operator features predict within-problem correctness above all span-free baselines and the LLM self-judge (the OST reaches 0.701 cross-dataset, matching the content-augmented Op-XGB classifier while reading only operator labels), and that the OST—trained once on full sequences—enables correctness estimation from partial traces at any depth, surpassing an Op-XGB upper bound retrained per depth. 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 [3]. 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 [12, 13, 18], operate on arbitrary syntactic features such as sentences [14, 15], or rely on domain-specific vocabularies [16]. 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 [10, 11].

Monitoring and scalable oversight. The scalable oversight community has extensively explored ways to characterize LLM behavior [9, 19, 20], and chain-of-thought monitoring has been explored as a prominent direction [8, 7]. 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 [21], but recent work has attempted to improve this in frontier models [7, 22]. 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 [23, 24], and mechanistic interpretability techniques [25]. Similar to

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

this line of work is that of process reward models [6] and outcome reward models [26], including work on verifiers in test-time scaling [27, 28]. This work has been particularly prolific in mathematics problems [26, 6, 29]. Other work has explored test time scaling techniques for self-verification, including SelfCheck [23] and others [30, 31, 32].

3 Methods

We now describe how we build an unsupervised mechanism to infer operators from reasoning traces across 12 LLMs. Our pipeline is driven by linguistic principles and requires no *a priori* definition of an operator vocabulary.

Data collection. We collect reasoning traces from 12 thinking language models spanning 6 families (Table 4 in Appendix A) on 8 benchmarks: MATH-500 [33], GPQA Diamond [34], AIME 2024, LiveCodeBench [35], HumanEval [36], MMLU-Pro [37], ARC-Challenge [38], and BIG-Bench Hard [39]. All models are queried with a 65,536-token context budget. Raw chain-of-thought tokens are collected via the OpenRouter API for all models except Claude, which is queried directly via the Anthropic API with extended thinking enabled; per-model API details are given in Appendix A. After removing truncated and malformed traces, we retain **44,662 traces** ($\approx$3,720 per model).

Pivot-based span segmentation. Each trace is segmented into **spans**. A span begins at a *pivot sentence*: any sentence whose first three alpha tokens (the *sentence start*) appear frequently enough across the corpus to be recognized as a discourse signal. Formally, $\mathrm{pivot}(s) = (w_1, w_2, w_3)$ where w_i are the ordered lowercase alphabetic tokens at the start of sentence s. A discourse pivot p is included if (1) it appears in ≥ 100 distinct traces (*frequency filter*), (2) it appears in traces from ≥ 3 distinct datasets (*domain-diversity filter*), and (3) all three tokens w_i belong to the top-2,000 most frequent corpus tokens (*vocabulary filter*).

These thresholds are set to balance specificity and generality. The frequency threshold of 100 traces is chosen to exclude idiosyncratic phrases that appear in only a handful of traces while retaining common discourse moves; lowering it to 50 adds many domain-specific phrases that are not discourse-functional. The domain-diversity filter of 3 datasets ensures that any accepted pivot spans multiple task types, preventing dataset-specific phrases from entering the vocabulary. The top-2,000 vocabulary filter ensures pivots are composed of common English function words rather than domain-specific content tokens (e.g., chemistry formulae, programming keywords); we verified that varying this between 1,500 and 3,000 does not qualitatively change the resulting clusters.

Semantic pivot embedding and clustering. We embed all discourse pivots using the `intfloat/e5-small-v2` sentence encoder [17]. Sentence-level embeddings capture full-phrase meaning, correctly separating pivots such as "let me think" and "let me verify" that share a common prefix. We cluster pivot embeddings with K-means ($K \in \{6, \ldots, 11\}$, 30 restarts), selecting K by maximizing Cohen's κ against an independent LLM judge on held-out spans. The optimal $K{=}7$ yields $\kappa = 0.693$–0.720 across three judges (Table 6, Appendix C).

Independence from correctness. The operator discovery pipeline uses no correctness labels: pivots are filtered by frequency and domain diversity, embeddings are fit on pivot text alone, and cluster assignments are by nearest-centroid lookup. Correctness labels enter only in the downstream prediction step (§6), where Op-XGB and the OST are trained in a standard supervised 5-fold CV protocol. The operators themselves are thus a fully unsupervised intermediate representation.

Computational cost. The discovery pipeline is lightweight. Pivot extraction (frequency counting over all sentence starts) takes under 3 minutes on a single CPU core for 44,662 traces. Embedding the 5,464 accepted pivots with `e5-small-v2` takes 6 seconds; K-means on 5,464 points takes 12 seconds. Annotating a new trace requires only a dictionary lookup per span (no embedding at inference time), adding $<$1 ms per trace. End-to-end operator discovery over the full corpus takes under 5 minutes, and annotating the entire 44K-trace corpus takes under 2 minutes.

4 Reasoning operators

Seven operators emerge from unsupervised clustering (Table 1; Figure 1). They span the space of discourse moves in extended reasoning: goal-setting, fact-anchoring, inference-drawing, exploration, hedging, self-correction, and constraint-specification. As a more coarse classification, operators can be broken into *committal operators* (INITIATING, INFERRING, CONSTRAINING, GROUNDING) and *reflective operators* (QUALIFYING, BACKTRACKING, HYPOTHESIZING). These operators recur across all 12 models and all 8 benchmarks, suggesting that extended reasoning—regardless of

Table 1: The 7 discovered reasoning operators. Representative pivots are the most frequent 3-tuples in each cluster.

Operator	Representative Pivots	Description
INITIATING	let me think, let me check, let me verify	Explicitly launches a new cognitive operation
QUALIFYING	hmm, but the problem, but let me	Introduces caveats or complications
GROUNDING	i need to, the question is, looking at	Anchors reasoning in facts or given information
INFERRING	so the answer, thus final answer, yes	Draws a conclusion from prior steps
HYPOTHESIZING	alternatively perhaps, for example if, so perhaps	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

training data, architecture, or problem domain—organizes itself into a common compositional vocabulary. Figure 1 shows representative span excerpts per operator across three of the eight benchmarks. An illustration of a full operator sequence in Figure 1 and Appendix J.

Evaluation with an LLM judge. We first test whether the discovered operators are semantically coherent from the perspective of independent judges. Each LLM judge receives only cluster names, descriptions, and the top 12 representative pivot phrases per cluster—no training spans. It labels 50 held-out spans per cluster ($n = 350$ total). Table 2 reports classification accuracy for three judges. All reach 70–76% on the 7-way classification task, 4.9–5.3$\times$ above the 14.3% chance baseline. Adding 3 example spans per cluster to the prompt raises accuracy from 69.8% to **74.6%**: few-shot examples aid pattern-matching from a closed 7-label set.

Cluster naming stability. To assess the semantic stability of the clusters, we re-ran the LLM naming step for the clusters by sampling 30 seeds of exemplars. For each name and resulting description, we computed pairwise cosine similarity of description embeddings across the seeds. The resulting descriptions were highly consistent within clusters with a median pairwise cosine similarity of 0.660, substantially above both an across-cluster baseline of 0.467 ($r_{\mathrm{rb}} = +0.72$) and a matched random-group control of 0.565 ($r_{\mathrm{rb}} = +0.39$; $p < 10^{-149}$). All seven operators showed significant within-cluster stability, indicating that the naming procedure recovers coherent and reproducible operator semantics; by contrast, randomly grouped spans did not elicit comparably coherent names. See Appendix K.1 for more details.

Table 2: LLM-judge validation (n=350 spans, 50 per cluster). Judges receive cluster names, descriptions, and top-12 representative pivots. **Exemplars**: prompt includes 3 example spans per cluster. Full κ values are reported in Appendix C.

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%**

5 Analysis of operator distributions

Operators show distinct patterns when stratified by dataset and model (Figure 2). Mathematical reasoning benchmarks (AIME, MATH) show higher GROUNDING and CONSTRAINING (step-by-step derivation), while commonsense benchmarks (ARC-C, BBH) use more GROUNDING and elevated HYPOTHESIZING. Challenging multiple choice datasets like GPQA and MMLU-Pro exhibit higher use of HYPOTHESIZING and QUALIFYING than any other dataset. Coding benchmarks (LiveCodeBench, HumanEval) exhibit elevated CONSTRAINING, consistent with specification-first programming. Models further show operator usage fingerprints that roughly cluster with model families, a phenomenon that's supported by the results in §6.1. R1-distill model profiles are both very similar, and Qwen models show almost identical operator proportions. Kimi/Moonshot models show high CONSTRAINING usage. Claude Haiku, Claude Sonnet, and Grok-3-Mini use excessive GROUNDING. Despite these differences, all 7 operators are represented in every model.

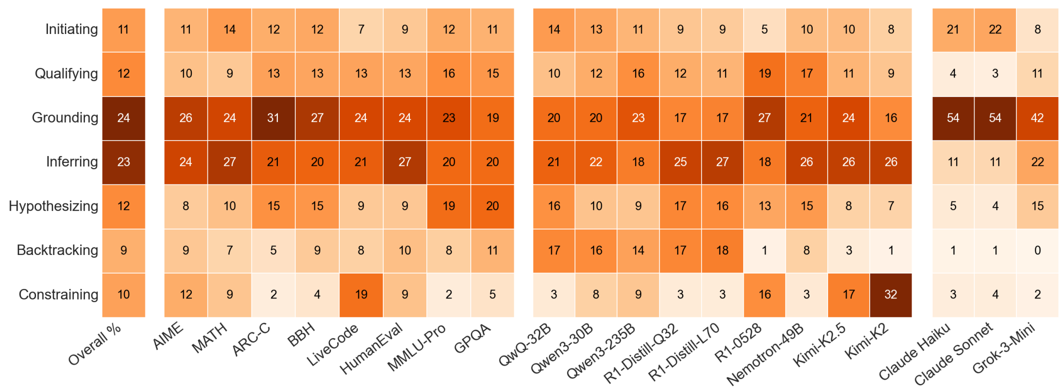

Figure 2: Distribution of operators across datasets (top) and models (bottom). Metrics are percent usage = average appearance of that operator for that given model/dataset vs. all other operators.

Transition structure. The operator transition matrix reveals Markov structure in reasoning: GROUNDING strongly self-transitions (sustained fact-anchoring), while BACKTRACKING most often leads into INITIATING (correction followed by a fresh attempt). HYPOTHESIZING frequently precedes INFERRING, consistent with a generate-and-test schema. Run-length analysis reinforces this: GROUNDING and CONSTRAINING accumulate long self-runs, while BACKTRACKING is almost always a single span—a brief interruption rather than a sustained state (Figure 11, Appendix L). We expand on this BACKTRACKING phenomenon in §5.2.

5.1 Operator usage on correct vs. incorrect traces

We see a few trends emerge in terms of operator usage on correct vs. incorrect traces (Figure 3). First, committal operators seem to be overrepresented in correct traces than in incorrect traces, with GROUNDING, INFERRING, INITIATING, and CONSTRAINING all being positive in correct - incorrect usage. 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. This may be indicative of correct traces showing more confident, forward-stepping operations rather than reflecting. This then led us to investigate these dynamics on easy vs. hard problems.

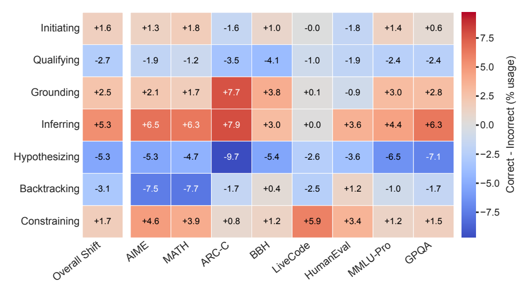

Figure 3: Shift of operator usage by model in correct and incorrect traces. % usage difference = take percent usage (Figure 2) and subtract usage in correct and incorrect traces.

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. For easy problems, correct traces exhibit significantly higher usage of committal operators than reflective operators, indicating confident forward-stepping through the problem. The usage of committal - reflective operators can be defined as the *committal-reflective gap*. 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). On average across all problems, committal usage peaks early and late in traces while reflective operators appear mid-trace (Figure 4a).

Hard problems show less obvious differences between correct and incorrect traces. 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 4a). 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

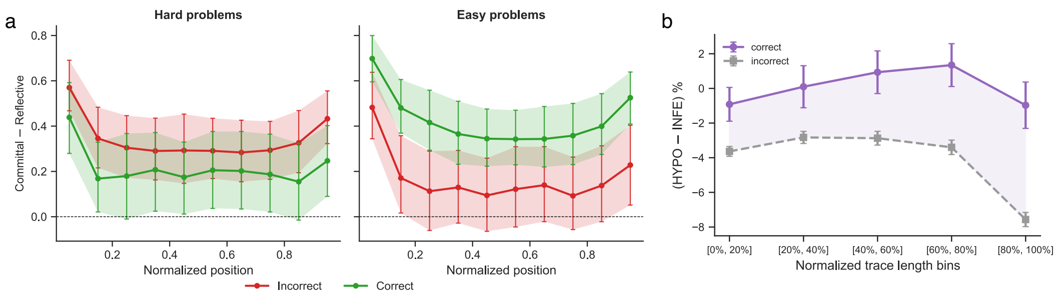

Figure 4: (a) Temporal distribution of committal - reflective operators for correct and incorrect problems. 95% CI shown at each normalized position. (b) HYPOTHESIZING- INFERRING gap over only hard problems for correct and incorrect traces. Shaded regions show the gap between correct and incorrect curves.

and reflective operators, yields some effects. When examining the distribution of these operators over time, the HYPOTHESIZING- INFERRING gap is particularly representative for correct vs. incorrect hard problems. 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 4b). This shows that reflective operators, specifically HYPOTHESIZING, might provide some benefit on hard problems while harming performance on easy problems.

5.2 Backtracking is a local operator

We centered the next analysis around BACKTRACKING and HYPOTHESIZING. 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 (re-checks a single calculation; method unchanged), SUB-PROBLEM (re-opens a specific case), or GLOBAL (abandons the current strategy), prompting the judge with the full reasoning trace and the event marked from the operator span to its next same-operator span. Local backtracking is explicitly defined as "The model is re-checking a single calculation, fact lookup, or specific claim...", and global event is defined as "model abandons or proposes to replace its current solution...". **BACKTRACKING and HYPOTHESIZING are overwhelmingly local revisions rather than strategy changes**, with statistically indistinguishable scope distributions: BACKTRACKING is 85.4% LOCAL / 12.8% SUB-PROBLEM / 1.6% GLOBAL ($n = 1{,}294$), and HYPOTHESIZING is 80.2% / 18.2% / 1.6% ($n = 192$). This is supported by literature that shows that backtracking is often superficial [40] and rarely changes a model's answer [41]. More details are in Appendix H.

6 Downstream Applications

We now demonstrate how operators discovered by ReasonOps can be used to predict downstream properties of reasoning traces, namely model identity and correctness of the trace.

6.1 Model Identification

If operators capture stable behavioral structure, traces from the same model family should cluster together in operator space. We train **Op-XGB**, an XGBoost classifier [42] (400 trees, max depth 6) on a curated operator feature set: a 117-dim handcrafted operator feature vector—global and quartile-localized operator frequencies, bigram transition matrix, run-length statistics, first/last operator one-hots, and entropy/length scalars—concatenated with an 8,000-feature anchor-phrase TF-IDF representation; the TF-IDF is re-fit inside each training fold (full feature definitions in Appendix E). Op-XGB is trained under problem-level 5-fold CV with model identity as the target. It achieves overall accuracy $= 79.9\%$ (chance $= 8.3\%$) and macro-AUC $= \mathbf{0.987}$. Per-model one-vs-rest AUCs (Figure 5) span 0.967 for R1-Distill-Llama-70B to 0.997 for Kimi-K2, Kimi-K2.5, and Grok-3-Mini, with the lowest AUCs concentrated in within-family pairs where intra-family confusion is hardest. The full 12-class confusion matrix (Figure 12, Appendix L.2) makes this explicit: R1-Distill-Llama-70B traces are predicted as R1-Distill-Qwen-32B at 0.41 and the reverse at 0.29 (the largest off-diagonal mass), consistent with their shared distillation lineage from DeepSeek-R1;

Table 3: Within-problem AUC (WP-AUC) for correctness prediction. **CD** = 5-fold problem-level CV trained on all datasets pooled; **ID** = 5-fold CV trained and evaluated within each dataset. Length, Backtrack, and Wait use logistic regression on a single monotonic feature; AUC is rank-invariant to the training fold (CD=ID). Op-XGB: 117-dim operator features (frequencies, quartile localization, bigram transitions, run lengths) + 8K anchor-phrase TF-IDF, concatenated; XGBoost (400 trees, depth 6). See Appendix E for full feature details. OST at 100% trace depth; see §6.3 for partial-trace results. **Bold** = best. Underline = 2nd best. Our methods.

	Op-XGB		OST		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

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).

6.2 Correctness Prediction

We next evaluate whether operator composition can be used to predict answer correctness. We focus on the 6 benchmarks where operator structure is most informative: AIME, ARC-Challenge, GPQA, LiveCodeBench, MATH-500, and MMLU-Pro. We exclude BIG-Bench Hard and HumanEval: BBH uses answer-matching formats where operator structure has near-chance predictive signal, and HumanEval correctness is determined by code execution rather than reasoning quality. We verify in §6 that this exclusion is empirically motivated. Our primary metric, *WP-AUC* (within-problem AUC; adapted from [43]), computes AUC separately for each problem's trace group and averages—isolating the ability to discriminate better from worse attempts on the *same* problem, with difficulty factored out. We use 5-fold problem-level CV. All traces for a given problem are assigned to the same fold, preventing leakage from multi-model evaluation.

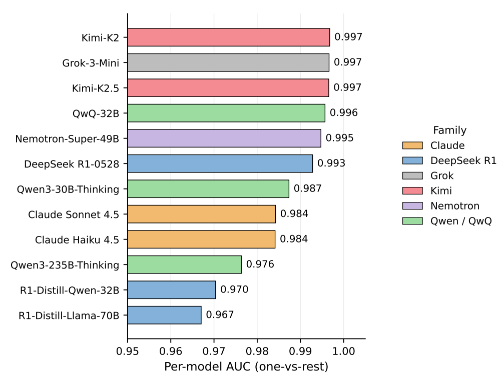

Figure 5: Per-model one-vs-rest AUC for the Op-XGB model identification classifier, colored by family. AUCs span 0.967 (R1-Distill-Llama-70B) to 0.997 (Kimi-K2, Kimi-K2.5, and Grok-3-Mini, tied); within-family pairs (R1-Distill, Claude, Kimi) sit at the lower end where intra-family confusion is concentrated.

We compare six methods: (1) **trace length** (log response character count), (2) **backtrack count** (fraction of spans labeled BACKTRACKING), (3) **wait count** (raw count of "wait"-prefixed spans), (4) **SelfCheck** [23]: the same model that generated the trace reads the problem and its full reasoning and predicts correctness without the gold answer—no training, (5) **Op-XGB**, and (6) **OST** (Operator Sequence Transformer; described in §6.3). Methods (1)–(3) use logistic regression on a single scalar feature with the same 5-fold problem-level CV protocol; since a 1-D logistic regression is a monotone transformation, AUC is fold-invariant (CD=ID). Method (4) requires no training. Methods (5)–(6) are evaluated under two protocols: **CD** (cross-dataset), trained on all datasets pooled, and **ID** (in-dataset), trained and evaluated within each dataset separately.

Table 3 reports WP-AUC per dataset for all methods.

Trace length is near chance globally (WP-AUC = 0.551 within-dataset). Backtrack count (0.594) and wait count (0.603) outperform length, confirming that reflective operator frequency carries sig-

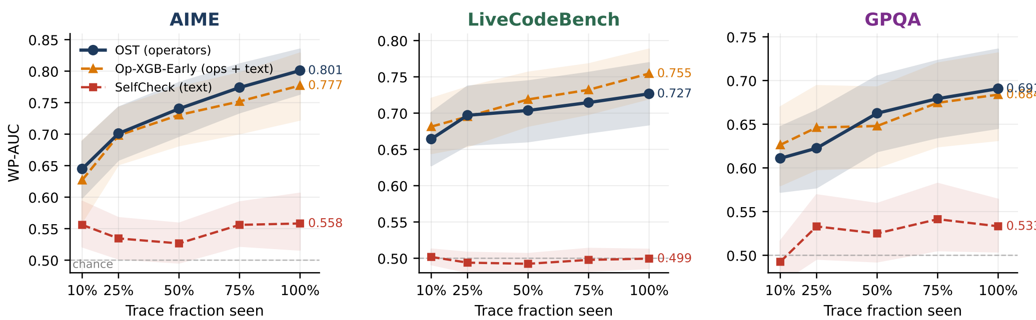

Figure 6: Correctness prediction WP-AUC at increasing trace depths for the three best-performing datasets (AIME, LiveCodeBench, GPQA). Navy solid (**OST**): operator sequence transformer trained once on full sequences, reading the discrete operator label sequence at any prefix length without retraining. Amber dashed (**Op-XGB-Early**): full Op-XGB recipe (117-dim operator features + 8,000-dim anchor TF-IDF; Appendix E), retrained from scratch at each depth as an upper-bound comparison. Red dashed (**SelfCheck**): the same model that generated the trace reads its own raw reasoning up to depth $p\%$ and predicts correctness with no training. Shaded bands: 95% CIs.

nal even without full sequence modeling. SelfCheck [23]—the same model that generated the trace reading its own reasoning and predicting correctness—reaches WP-AUC $= 0.512$ globally (near chance), a striking null result: reasoning models are overconfident and predict "correct" for both correct and incorrect traces at similar rates. Op-XGB reaches 0.701 cross-dataset and 0.723 within-dataset globally, with strong within-dataset performance on AIME (0.838 ID) and LiveCodeBench (0.795 ID). The OST reaches 0.701 cross-dataset and 0.703 within-dataset at full trace depth—tied with Op-XGB cross-dataset despite reading only operator labels, with no access to anchor phrase text. OST surpasses Op-XGB on ARC-Challenge (0.645 vs. 0.618 CD) and MATH (0.662 vs. 0.570 ID), where temporal operator dynamics capture structure that aggregate feature counts miss.

6.3 Early Correctness Prediction

A practical question is whether correctness can be estimated from a partial trace, before the model finishes reasoning. We introduce the **Operator Sequence Transformer** (OST): a lightweight Transformer encoder ($\sim$800K parameters; 4 layers, d=128, 4 heads, pre-LayerNorm) trained end-to-end directly on operator sequences with no pretraining. Each operator token is represented as the sum of a unigram embedding, a *bigram* (transition) embedding encoding the previous$\to$current operator pair, and a continuous sinusoidal position encoding proportional to the token's normalized position within the visible prefix—giving the model a fine-grained sense of *where* in the trace each operator appears. The model is trained with a pure contrastive loss that directly encourages correct traces to score higher than incorrect traces on the same problem, under the same 5-fold problem-level CV protocol as the other methods. Crucially, the OST is *trained once on full operator sequences*; at inference time it accepts any operator-sequence prefix without retraining, making it a practical single-model early-prediction system deployable at any trace depth. More details are in Appendix D.

Figure 6 shows OST WP-AUC vs. trace depth for the three best-performing datasets. On AIME, WP-AUC rises from 0.645 at 10% depth to 0.740 at 50% and 0.801 at 100%—at half the trace it already surpasses all span-free baselines. LiveCodeBench and GPQA follow similar trajectories, reaching 0.727 and 0.691 at full depth. Globally, the OST rises from WP-AUC $= 0.605$ at 10% depth to 0.664 at 50% and 0.701 at 100%, monotonically improving with trace length.

We compare against **Op-XGB-Early**, which applies the same Op-XGB feature recipe at each depth d: we truncate every trace to its first $d\%$ of spans, recompute the operator-sequence statistics and the TF-IDF on the truncated span set, and fit a fresh XGBoost from scratch. Per-depth retraining gives Op-XGB-Early an explicit upper-bound advantage—it is re-optimized for each truncation level—whereas the OST is trained only once on full sequences. Globally, Op-XGB-Early reaches WP-AUC 0.620 at 10%, 0.647 at 25%, 0.661 at 50%, 0.682 at 75%, and 0.699 at 100%. The OST and Op-XGB-Early track each other within a few thousandths across all depths: OST$-$Op-XGB-Early gaps are -0.015, -0.016, $+0.003$, $+0.006$, and $+0.003$ at depths $\{10, 25, 50, 75, 100\}\%$. The single OST model thus matches a per-depth-retrained Op-XGB—which has access to both the operator features *and* the raw anchor text—for partial-trace prediction at 50% depth and beyond, despite reading only the discrete operator label sequence and never being retrained for partial inputs.

By contrast, the partial-trace SelfCheck baseline—the same model reading its own first $p\%$ of raw reasoning text—is near chance at all depths (0.50–0.56 across datasets), confirming that the OST's signal comes from the structural operator representation rather than from text content alone.

7 Discussion

ReasonOps is an unsupervised method for annotating reasoning traces, built from careful observations of their semantic structure. Three-token pivots strike a useful balance for discourse analysis: they disambiguate the polysemy of single-token markers like "but" or "so" (e.g., distinguishing "wait," "wait no," and "wait let me" as backtracking, rejection, and correction respectively) 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 (e.g., "the double bond" in chemistry traces) that would otherwise contaminate the pivot set with non-discourse-functional content. ReasonOps is a robust method to provide domain-agnostic, model-agnostic annotations of reasoning traces that enable universal analysis and downstream tasks such as model identification and correctness prediction.

Future work and limitations. 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 more useful for agentic traces where actions may change drastically based on the harness. The operators define a structured vocabulary for step-level correctness prediction without human annotation. OST's early correctness signal can trigger intervention (e.g., beam expansion, early stopping) at any trace position, enabling compute-adaptive generation. There are also several limitations of ReasonOps. First, the pipeline relies on sentence-initial pivots; spans bounded at a sub-sentence level cannot be labeled. Second, the operators are validated by LLM judgment, which introduces its own biases. Third, cross-model transfer is limited: operator frequency features partially capture model-specific patterns that do not fully generalize. Finally, the corpus covers 8 benchmarks in English; generalization to other languages is not verified.

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.

Summary. Seven reasoning operators emerge from unsupervised analysis of 44,662 chain-of-thought traces across 12 thinking LLMs and 8 benchmarks. Three independent LLM judges confirm the taxonomy (70–76% classification accuracy; chance: 14%). Operator sequences identify the source model with macro-AUC $= 0.987$. Structural features predict within-problem correctness well above all span-free baselines and above the LLM self-judge (0.512, near chance). The OST reaches WP-AUC $= 0.701$ cross-dataset—tied with Op-XGB (0.701 CD) while reading only operator labels, not the additional text features Op-XGB receives. The OST predicts correctness monotonically from partial traces, reaching 0.664 at 50% trace depth and surpassing an Op-XGB baseline retrained at each depth, with no per-depth retraining required. The convergence of architecturally diverse models on the same 7 discourse-level moves suggests that extended reasoning organizes itself into a common compositional vocabulary.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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*, 2025.

[13] 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.

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

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

[16] 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*, pages 18278–18299, 2025.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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*, pages 15012–15032, 2024.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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)*, pages 9426–9439, 2024.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

[39] 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*, pages 13003–13051, 2023.

[40] 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.

[41] 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.

[42] 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*, pages 785–794, 2016.

[43] 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*, pages 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	S[1]
Qwen3-30B-Thinking	Qwen	30B (3B act.)	MoE S[1]
Qwen3-235B-Thinking	Qwen	235B (22B act.)	MoE S[1]
DeepSeek-R1-0528	DeepSeek	671B MoE	S[3]
R1-Distill-LLaMA-70B	DeepSeek	70B	Distilled S[3]
R1-Distill-Qwen-32B	DeepSeek	32B	Distilled S[3]
Claude-Sonnet-4.5	Anthropic	—	Extended thinking S[4]
Claude-Haiku-4.5	Anthropic	—	Extended thinking S[4]
Grok-3-Mini	xAI	—	S[5]
Nemotron-Super-49B	NVIDIA	49B	S[6]
Kimi-K2-Thinking	Moonshot	MoE	S[7]
Kimi-K2.5	Moonshot	MoE	S[2]

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, HumanEval, and ARC-AGI-2 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 trace count is lower because AIME 2024 has only 30 problems.

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 2024	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	κ
6	0.604
7	**0.693**
8	0.611
9	0.648
10	0.573
11	0.558

D Operator Sequence Transformer (OST) Architecture

The OST is a lightweight Transformer encoder trained end-to-end on operator token sequences. Its design is optimized for long sequences of discrete operator labels (typical length: 50–500 tokens) while remaining parameter-efficient enough to train in 5-fold CV.

Tokenization. Each span in a trace is assigned one of 7 operator labels (integer indices 0–6). The OST processes the sequence of these labels for a trace prefix of specified depth.

Input embeddings. Each position t in the operator sequence receives a 128-dimensional input vector formed as the sum of three components:

- **Unigram embedding**: a learned embedding $E_u[o_t] \in \mathbb{R}^{128}$ for the current operator o_t;
- **Bigram (transition) embedding**: a learned embedding $E_b[o_{t-1}, o_t] \in \mathbb{R}^{128}$ for the previous→current operator pair (special "start" token for $t = 0$);
- **Continuous sinusoidal position encoding**: standard sinusoidal encoding scaled by the *normalized* position t/T within the visible prefix, giving the model a fine-grained sense of where in the trace each operator appears.

Transformer body. The model uses a standard pre-LayerNorm Transformer encoder with 4 layers, $d_{\text{model}} = 128$, 4 attention heads, feed-forward width $= 512$, and 5% token dropout during training. Attention is full (not causal), since the model reads a complete prefix.

Pooling and scoring. Sequence representations are aggregated by attention pooling (learned query vector) into a single 128-dimensional vector, followed by a two-layer MLP ($128 \to 64 \to 1$) producing a real-valued correctness score. Total parameter count: ≈800,000.

Training. The OST is trained with a pure contrastive loss: for every problem with at least one correct and one incorrect trace in the training fold, all pairs (i^+, i^-) of correct and incorrect traces for that problem are formed, and the loss pushes the correct trace to score higher. This directly optimizes the within-problem ranking that WP-AUC measures. Training uses AdamW ($\eta = 3 \times 10^{-4}$, weight decay $= 0.01$), batch size 64, and early stopping on fold validation WP-AUC.

Early prediction. At inference time the OST is given only the first $p\%$ of operator tokens (by span count), allowing evaluation at any trace depth. The sinusoidal position encoding is re-normalized to the partial prefix length.

E Op-XGB Feature Details

Op-XGB combines a 117-dimensional handcrafted operator-sequence feature vector with an 8,000-feature anchor-phrase TF-IDF representation. Both feature sets are derived from the operator labels and sentence-initial pivot phrases—no substantive math, code, or reasoning content of the trace is used.

Operator feature vector (117 dim). For each trace's operator sequence $\sigma = (\sigma_1, \ldots, \sigma_n) \in \{0, \ldots, 6\}^n$:

- **Global frequencies** (7): the share of each operator across the full sequence, $\mathrm{freq}_k = \frac{1}{n}\sum_t \mathbf{1}[\sigma_t = k]$.
- **Quartile frequencies** ($7 \times 4 = 28$): the share of each operator within each of four equal-sized contiguous chunks of σ, capturing temporal localization.
- **Scalar summaries** (5): Shannon entropy of the global frequency vector, the maximum of freq, the number of distinct operators that appear, $\log(1 + n)$, and the immediate-repeat rate $\frac{1}{n-1}\sum_{t\geq 2} \mathbf{1}[\sigma_t = \sigma_{t-1}]$.
- **First/last one-hot** ($7 + 7 = 14$): one-hot encodings of σ_1 and σ_n.
- **Bigram transition matrix** (49): the 7×7 matrix of bigram frequencies $\frac{1}{n-1}\sum_{t\geq 2} \mathbf{1}[\sigma_{t-1}{=}a, \sigma_t{=}b]$, flattened.
- **Run-length statistics** ($7 + 7 = 14$): for each operator k, the mean and max length of consecutive runs of k, each normalized by n.

This vector is computed once per trace and depends only on the discrete operator label sequence; it carries no anchor text.

Anchor-phrase TF-IDF (8,000 dim). We additionally extract each span's anchor (the sentence-initial pivot phrase, truncated to 80 characters), concatenate all anchors of a trace with single spaces, and fit `sklearn.feature_extraction.text.TfidfVectorizer` with `max_features=8000` and `sublinear_tf=True` on the training-fold anchor strings (default unigram tokenization, sublinear TF $1 + \log(\mathrm{tf})$, IDF reweighting, vocabulary re-fit inside each fold to prevent leakage). The anchor strings consist only of discourse-level pivot phrases (e.g., "let's denote", "so we have", "wait, that means")—they are exactly the inputs that the unsupervised operator-discovery pipeline embeds and clusters into the 7 operators.

Classifier. The 117-dim operator feature vector and the 8,000-dim TF-IDF vector are concatenated to form an 8,117-dim per-trace representation, which is passed to XGBoost (400 trees, max depth 6, learning rate 0.05, subsample 0.8, colsample_bytree 0.8, logloss objective). The classifier is trained with problem-level 5-fold CV (cross-dataset) or within each dataset separately (in-dataset). XGBoost predicted probabilities are used directly as scores for WP-AUC computation. Op-XGB therefore reads operator information at *both* levels of abstraction (discrete labels and the raw pivot phrases that produce them), in contrast to the OST, which reads only the discrete labels.

Op-XGB-Early variant. For the early-prediction comparison (§6.3, Figure 6), at each depth $d \in \{10, 25, 50, 75, 100\}\%$ we truncate each trace's spans to the first $\lceil N \cdot d/100 \rceil$ spans, recompute the 117-dim operator feature vector on the truncated operator subsequence, recompute the anchor TF-IDF on the truncated anchor text, and re-fit a fresh XGBoost on the concatenated representation under the same 5-fold problem-level CV. This per-depth retraining provides an upper bound on what the full Op-XGB feature recipe can achieve when the model is re-optimized for each truncation level—in contrast to the OST, which is trained once on full sequences and evaluated on partial prefixes without retraining.

Op-XGB for model identification. For 12-class source-model identification (Figure 12), we use the identical Op-XGB feature recipe described above (117-dim operator features + 8,000-dim TF-IDF, concatenated to an 8,117-dim vector) and train XGBoost as a multi-class classifier (`multi:softprob`, same hyperparameters as the binary version) under the same problem-level 5-fold CV protocol. Reusing the full Op-XGB recipe ensures that the same model class supports both correctness prediction and model identification.

F Operator Usage by Correctness per Benchmark

Figure 7 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: $\sim$24% incorrect vs. $\sim$14% correct for HYPOTHESIZING). On LiveCodeBench, CONSTRAINING is 10 pp higher in correct traces.

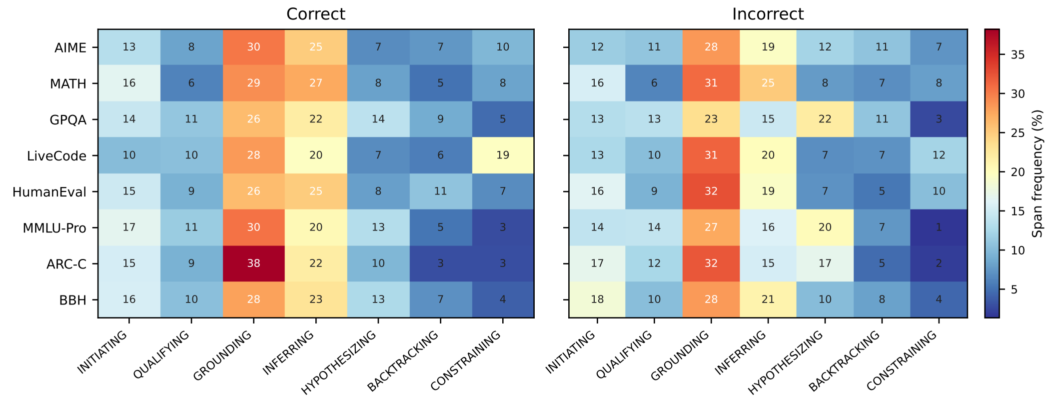

Figure 7: Operator frequency (%) per benchmark, split by correctness. Left: correct traces. Right: incorrect traces.

G Temporal Operator Profiles

Figure 8 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—a signature of failed exploration not followed by recovery.

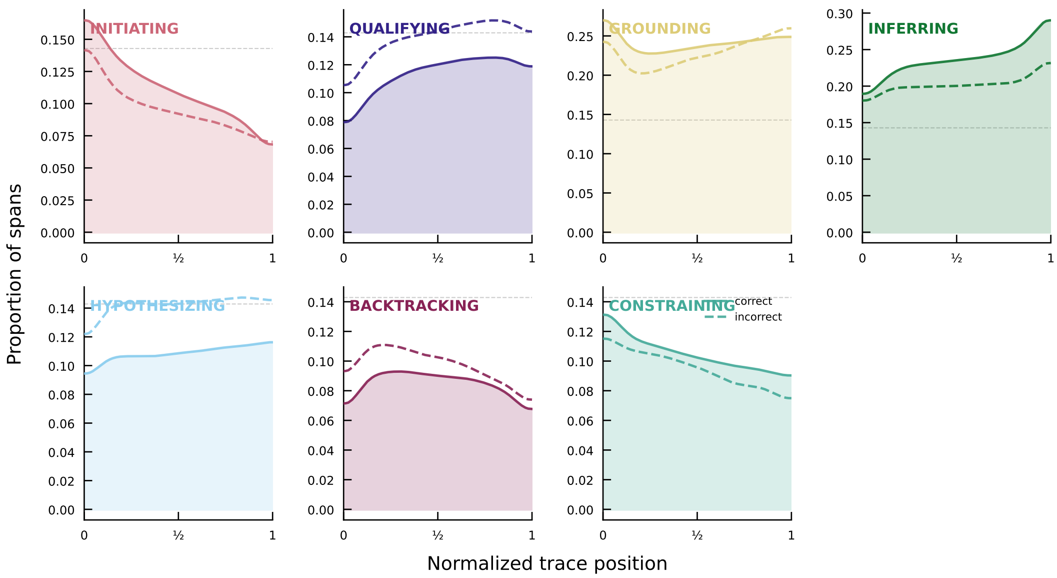

Figure 8: Operator density as a function of normalized trace position (Gaussian KDE). Dashed reference = uniform (1/7). Correct: solid; incorrect: dashed.

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 sees the GPQA problem and the full reasoning trace with the target event marked: a contiguous region from the target operator span to the next span of the same operator class (or trace end). The system rubric is reproduced below; the judge returns a JSON object `{"scope": ..., "rationale": ...}`. See prompt below for more information.

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 9 shows WP-AUC (cross-dataset protocol) across all six baselines for each of the 6 benchmarks. BIG-Bench Hard and HumanEval are excluded (see §3). Op-XGB and OST dominate on math-heavy and code benchmarks; the LLM self-judge is near chance on all datasets.

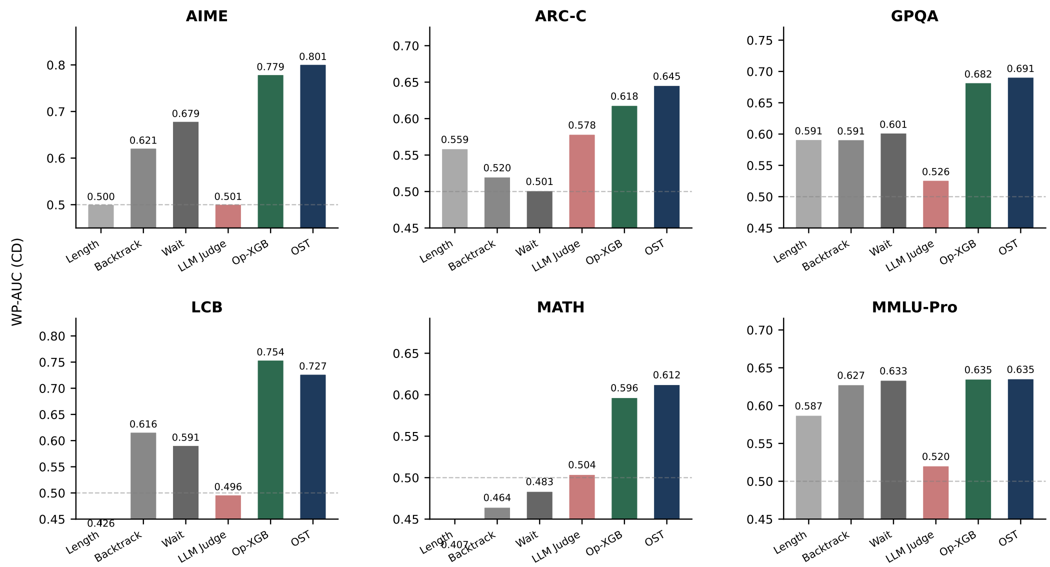

Figure 9: Per-benchmark WP-AUC (cross-dataset CV) for all six methods. Gray bars: span-free baselines (Length, Backtrack, Wait). Red bar: LLM self-judge. Green bar: Op-XGB (operator n-gram counts + anchor-phrase TF-IDF; XGBoost). Blue bar: OST (Operator Sequence Transformer).

J Labeled Trace Example

Table 8 shows a representative labeled trace from R1-Distill-LLaMA-70B on a MATH intermediate algebra problem (incorrect). Each span begins at a pivot sentence (anchor, upright font) followed by continuation sentences in *italics*.

K Evaluation

K.1 Stability Analysis

Methodology. To assess whether the induced operator labels reflect stable semantic structure rather than arbitrary exemplar choice, we held the $K = 7$ cluster assignments fixed and re-ran the LLM naming step (`claude-sonnet-4-6`, $T = 0$; same prompt as the discovery pipeline—top-12 anchor n-grams plus 5 example spans per cluster) across 30 exemplar seeds. The top n-grams are deterministic per cluster; only the 5 exemplars vary by seed, isolating naming stability from the upstream clustering. Each of the $30 \times 7 = 210$ resulting descriptions was embedded once with `text-embedding-3-small` (1536-d, L_2-normalized), and we computed pairwise cosine similarity as the inner product. We then formed three pair distributions: a *within-real* distribution (all $\binom{30}{2} = 435$ cross-seed pairs per cluster, 3,045 pairs pooled across the seven operators); an *across-real baseline* of 2,000 random pairs (i, j) with $\text{cluster}(i) \neq \text{cluster}(j)$ from the same description pool; and a *random-group control* generated by the identical naming protocol applied to seven disjoint "fake clusters" of 2,000 spans each, drawn uniformly from the full type-A span pool—these mix every real operator in corpus-rate proportions, so their top n-grams concentrate on high-frequency meta-phrases ("now we need", "let me think") with no operator-specific signal.

Statistical comparisons. We test whether the within-real distribution stochastically dominates each of the two reference distributions using a one-sided Mann–Whitney U test (alternative =

Table 8: Labeled trace with full span content. Model: R1-Distill-LLaMA-70B. Dataset: MATH. Correct: False.

Problem: Find all integer roots of $x^4{+}5x^3{+}9x^2{-}x{-}14{=}0$.

Operator	Span content
INFERRING	Okay, so I need to solve this equation and find all the integer roots. *And the task is to find all the integer roots.*
HYPOTHESIZING	Hmm, I remember that for polynomials the Rational Root Theorem can be helpful.
INITIATING	Let me try to recall how that works. *The Rational Root Theorem states that any rational root p/q has $p \mid a_0$ and $q \mid a_n$.*
HYPOTHESIZING	Therefore, the possible integer roots are the factors of -14.
INITIATING	Let me start with $x{=}1$ and $x{=}{-}1$ first.
GROUNDING	Substituting: $1{+}5{+}9{-}1{-}14{=}0$. *So $x{=}1$ is a root.*
INFERRING	So $x{=}{-}2$ is another integer root.
BACKTRACKING	Wait, but let me double-check my calculations because sometimes I might make an arithmetic mistake.
GROUNDING	Starting with $x{=}1$: $1{+}5{+}9{-}1{-}14{=}0$. Correct.
BACKTRACKING	But wait, the equation is a quartic, so it should have four roots in total.
QUALIFYING	But according to our tests, the other candidates didn't work, so perhaps the other roots are non-integer.
HYPOTHESIZING	Alternatively, maybe I missed something, so let me try another approach.
CONSTRAINING	Now, we have factored the polynomial as $(x{-}1)(x^3{+}6x^2{+}15x{+}14)$.
QUALIFYING	Therefore, the only integer roots are 1 and -2.

greater). For interpretability we report the rank-biserial effect size $r_{\text{rb}} = 2U/(n_1 n_2) - 1$, which rescales U to $[-1, +1]$ and equals $2\,P(A > B) - 1$: with sample sizes in the thousands the U-test rejects for trivial gaps, so r_{rb} is the load-bearing summary of *how much* the distributions are separated. The 435 within-cluster pair cosines per cluster are not statistically independent—they share the 30 underlying seed embeddings—which makes the U-test p-values anticonservative; r_{rb} does not depend on this independence assumption and is therefore reported as the primary result.

Results and interpretation. Figure 10 shows the three cosine-similarity distributions. The within-real pooled median cosine is 0.660 versus 0.467 for the across-real baseline and 0.565 for the random-group control. Real-cluster names dominate the across-cluster baseline with $r_{\text{rb}} = +0.72$ and the random-group control with $r_{\text{rb}} = +0.39$ (one-sided U-test $p \approx 0$ and $p < 10^{-149}$, respectively): roughly 86% of within-real pairs are more similar than a random across-cluster pair, and 69% more similar than a random within-fake-group pair. Every individual operator clears the across-cluster baseline (per-cluster U-test $p < 10^{-12}$; $r_{\text{rb}} \in [+0.22\ (\text{GROUNDING}, \text{weakest}),\ +1.00\ (\text{BACKTRACKING}, \text{strongest})]$). The random-group control falls midway between within-real and across-cluster ($r_{\text{rb}} = +0.44$ of random vs. across; median gap $= 0.098$): LLM descriptions of mixed-operator span pools are somewhat self-consistent, but far below the within-real ceiling, confirming that the coherence of real operator names is content-driven rather than a generic tendency of the naming prompt to produce similar-sounding sentences.

L Additional Analyses

L.1 Transition Structure and Run-Length Statistics

Figure 11 shows the operator transition probability matrix and run-length statistics. GROUNDING has the longest self-runs, reflecting sustained fact-anchoring that persists for multiple consecutive spans. BACKTRACKING is nearly always a single span—it interrupts but does not persist—which is consistent with its function as a brief error-signal rather than a sustained strategy change. The HYPOTHESIZING $\rightarrow$ INFERRING pathway has elevated probability relative to most other off-diagonal

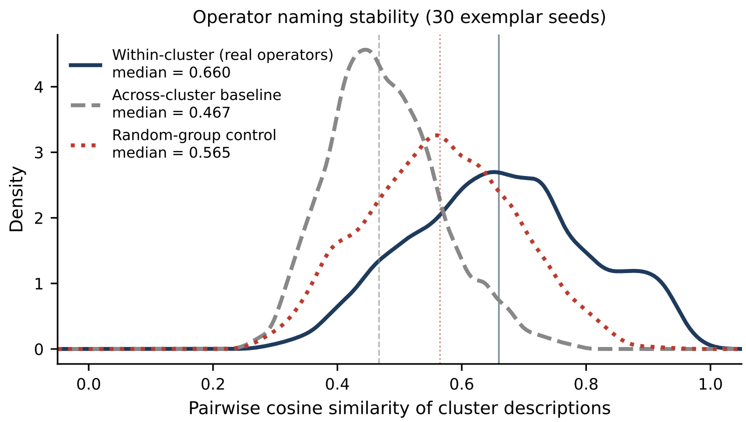

Figure 10: Naming stability: pairwise cosine similarity distributions between cluster-description embeddings generated under 30 exemplar seeds (`text-embedding-3-small`, L_2-normalized). The *within-cluster* distribution (median $= 0.660$) is well separated from both the *across-cluster* baseline (median $= 0.467$) and the *random-group control* (median $= 0.565$), which applies the same naming protocol to seven disjoint random span groupings. The gap confirms that operator names are stable at the level of meaning and are not an artifact of the LLM prompt.

entries, consistent with a generate-and-test schema in which tentative scenarios are followed by conclusions.

L.2 Model Identification Confusion Matrix

Figure 12 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 AUCs span 0.967 (R1-Distill-Llama-70B) to 0.997 (Kimi-K2, Kimi-K2.5, Grok-3-Mini, all tied at this rounding).

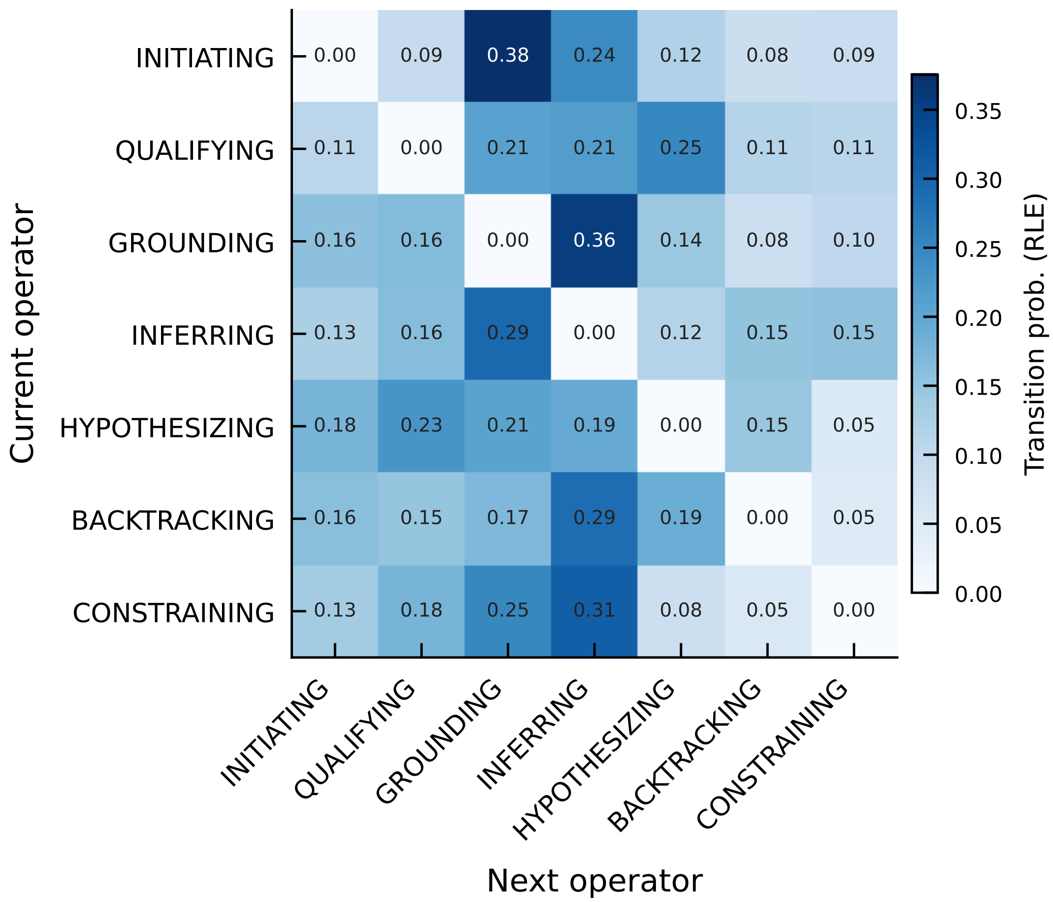

Figure 11: Left: operator transition probability matrix. Entry (i, j) = probability that operator j follows i. Right: run-length statistics per operator—mean and distribution of consecutive same-operator span counts. GROUNDING has the longest self-runs (sustained fact-anchoring); BACKTRACKING is nearly always a single span.

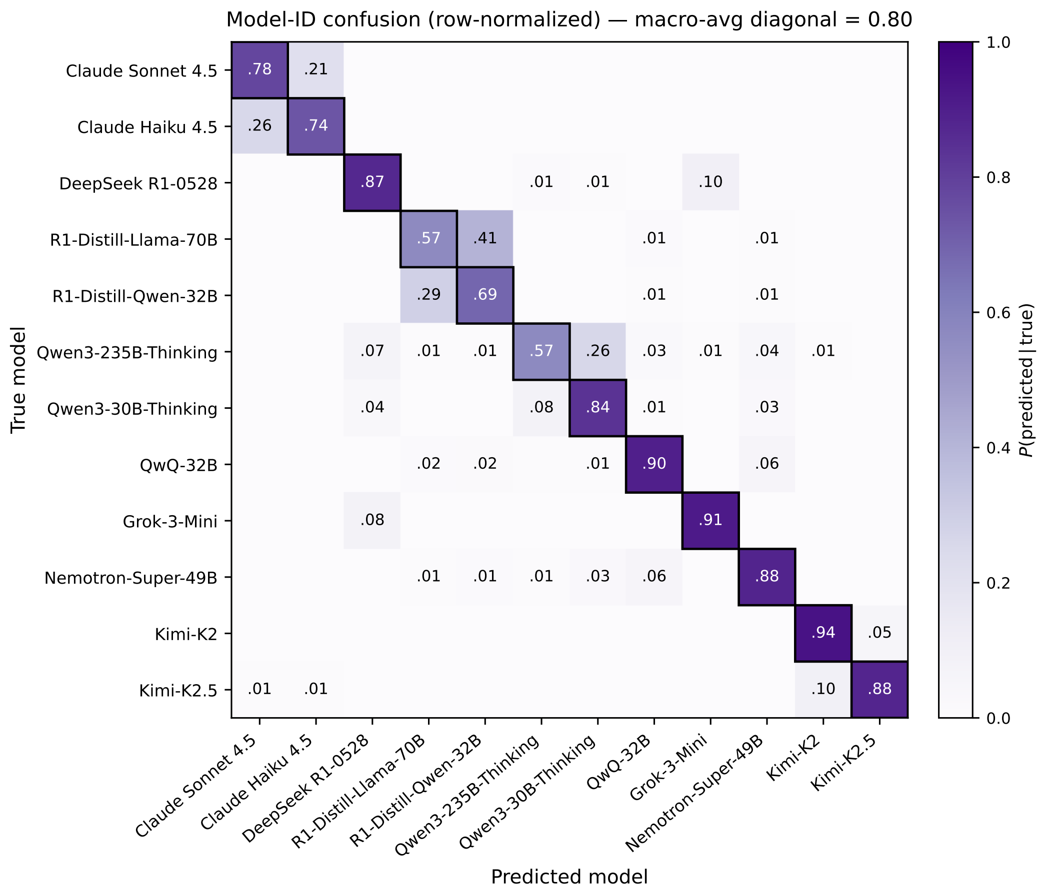

Figure 12: Model identification confusion matrix (row-normalized) under problem-level 5-fold CV with the full Op-XGB feature recipe (Appendix E). Accuracy = 0.799, macro-AUC = 0.987. Errors concentrate within model families (R1-Distill pair, Claude pair, Kimi pair).

Appendix References

[S1] Qwen Team. QwQ-32B: Embracing the power of reinforcement learning. `https://qwenlm.github.io/blog/qwq-32b/`, mar 2025. Blog post, March 6, 2025.

[S2] 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.

[S3] 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.

[S4] Anthropic. Introducing Claude 4. `https://www.anthropic.com/news/claude-4`, May 2025. Blog post, May 22, 2025.

[S5] xAI. Grok 3 beta — the age of reasoning agents. `https://x.ai/news/grok-3`, February 2025. Blog post, February 19, 2025.

[S6] 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.

[S7] Kimi Team. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[S8] 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.