Correct Prediction, Wrong Steps? Consensus Reasoning Knowledge Graph for Robust Chain-of-Thought Synthesis

Anonymous ACL submission

Abstract

Large language models (LLMs) have become increasingly used for various tasks, often coupled with chain-of-thought (CoT) prompting to boost accuracy. Recent work has shown that high label-prediction accuracy does not guarantee correct intermediate reasoning, and the causes of *reasoning flaws* vary from sample to sample, yet existing remedies either focus on a single domain or assume that one flaw type applies uniformly across samples. A simple mitigation method is to provide the model with the correct answer, but we show that this yields no consistent improvement in reasoning quality. This indicates that the problem cannot be fixed by LLMs' awareness of answers, and must instead be addressed through the *structure* of reasoning. Motivated by this, we propose CRAFT (Consensus Reasoning-knowledge-graph Aggregation for Flaw-aware Trace synthesis), which aggregates the consensus components shared across multiple candidate reasoning traces to synthesize improved ones. CRAFT consistently improves label-prediction accuracy on both logical and mathematical reasoning benchmarks, outperforming most baselines, while its post-processed traces achieve higher quality under fine-grained benchmark evaluation. Code and dataset are available at `https://anonymous.4open.science/r/CRAFT-2712/`.

1 Introduction

LLMs show impressive reasoning ability in logical (Zhou et al., 2025) and mathematical domains (Shao et al., 2024), increasingly trained via Reinforcement Learning that distinguishes good traces from bad (Lightman et al., 2023). These capabilities have been adopted across high-stakes domains like financial analysis (Zhao et al., 2024), medical treatment (Savage et al., 2023), and legal consultation (Yu et al., 2022). In such applications, what matters is not only the final answer an LLM outputs, i.e., its *label-prediction accuracy*, but also the *quality of its reasoning trace*, i.e., whether the intermediate steps leading to that answer are themselves correct. Conventionally, the latter is taken for granted: a correct answer is read as evidence of correct reasoning.

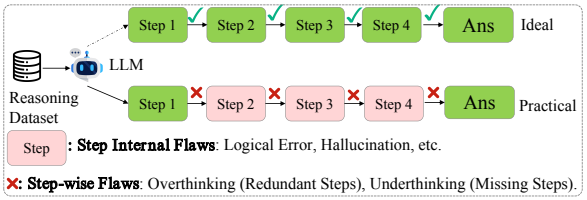

Figure 1: Problem background. LLMs' correct label-prediction is related to, but not fully determined by, whether reasoning steps are correct. Case studies of logical and mathematical reasoning are in Appendix I.

However, this assumption does not always hold: correct label-prediction is necessary but not sufficient evidence of correct reasoning. For instance, self-reflection based on previously generated traces can result in accuracy drops (Ling et al., 2025a,b), and by optimizing only for the final answer, RL can favor reasoning steps that reach the correct prediction through invalid intermediate logic (Yue et al., 2025). Figure 1 visualizes this: whereas an ideal trace has every step correct, in practice a trace can reach the correct answer despite flawed intermediate steps. Such flaws fall into two types: *Step Internal Flaws*, where an individual step is itself wrong (e.g., a logical error or hallucination), and *Step-wise Flaws*, where steps may each look valid but are mis-organized—omitting necessary steps (Underthinking) or piling on redundant ones (Overthinking). This poses risks for (1) LLM Distillation: models may be fine-tuned on traces generated by other LLMs; and (2) Dataset Annotation: such errors may propagate into benchmarks that rely on LLM-generated traces to annotate problem-solving methods.

Existing mitigation efforts fall short on two as-

pects. **(i) Reasoning Domain coverage:** Xu et al. (2026) directly targets this mismatch addresses only logical reasoning. As it is tailored to its specific structure, it does not transfer to other domains such as mathematical reasoning, even though the same reasoning flaws arise there. **(ii) Flaw-type coverage:** each method assumes a single flaw type holds for all traces: some attribute flaws to redundant steps (Overthinking) (Yu et al., 2025), others to missing steps (Underthinking) (Xu et al., 2025), yet in practice different samples trigger different flaws, yielding a mixture that no single-type assumption can capture.

Before proposing a problem-mitigation method, we first ask why these reasoning flaws arise. A natural hypothesis is that they arise from the model's uncertainty about the correct answer; if so, the simplest fix would be to provide the answer and ask the model merely to justify it. We test this hypothesis on two benchmarks, (1) PRM-Bench (Song et al., 2025): evaluating an LLM as a step-by-step verifier for detecting error steps, and (2) ROSCOE (Golovneva et al., 2023): evaluating reasoning-trace quality. Across four LLMs, providing the correct answer does not improve reasoning in most settings (Section 3.1), indicating that the flaws are structural rather than a matter of answer-awareness.

Since LLMs' awareness of answers does not help, the flaws lie in the reasoning trace itself, not the final answer. Our key observation is that flawed steps tend to be sample-specific, whereas correct ones recur across independently sampled traces (Wang et al., 2023). This motivates mitigating both flaw types via *cross-trace consensus*: aggregating correct reasoning steps of multiple candidate traces, while flawed ones are dropped out. To apply consensus to step content (catching *Step Internal Flaws*) and step structure (catching *Step-wise Flaws*), we propose **CRAFT** (Consensus Reasoning knowledge graph Aggregation for Flaw-aware Trace synthesis), a unified framework built on a *Reasoning Knowledge Graph* (RKG) that operates in three modules: **(I) Multi-Trace Generation & Steps Filtering**: roll out K candidate traces per sample, identify consensus terms via TF-IRF weighting, and remove structurally different steps through z-score filtering; **(II) Consensus RKG Construction**: convert traces into per-trace RKGs (steps as nodes, logical relations as edges), aggregate them into a consensus RKG G^*, and filter minor edges and nodes; **(III) Topology-guided Trace Synthesis**: generate one high-quality trace by traversing G^* in topological order.

To summarize, our contributions are three-fold:
▷ A benchmark study showing that providing correct answers yields no consistent improvement in reasoning, which indiates that the flaws are structural rather than a matter of answer-awareness.
▷ We propose CRAFT, a unified framework that jointly mitigates *Step Internal Flaws* and *Step-wise Flaws*, yielding consistent label-prediction accuracy gains across both logical and mathematical reasoning datasets.
▷ We further show that CRAFT's post-processed traces achieve higher quality under fine-grained benchmark evaluation.

2 Methodology

2.1 Correct Answer Guidance Study

A natural first question is whether flaws in LLM reasoning arise from uncertainty about the correct answer. If so, the simplest fix would be to provide the answer directly, and merely require the model to justify the given answer. We test this hypothesis on two complementary benchmarks: (1) PRM-Bench (Song et al., 2025): benchmarking an LLM as a step-by-step verifier for identifying erroneous steps, and (2) ROSCOE (Golovneva et al., 2023): benchmarking reasoning trace quality generated by an LLM. Experiments are carried out across two settings: (1) **w/ Answer**: the correct answer is provided in the prompt, and (2) **w/o Answer**: no correct answer is provided.

The results (Section 3.1) are counterintuitive: In general, providing the correct answer yields no consistent improvement in reasoning. This rules out answer uncertainty as the cause of reasoning flaws, pointing to a deeper issue: flaw types are heterogeneous across samples; when LLMs reason over a dataset, the result can be a mixture of flaw types: some traces suffer from overthinking, others from underthinking, and still others from step-internal flaws, all generated within the same dataset. A framework must mitigate both *Step Internal Flaws* and *Step-wise Flaws*.

2.2 Proposed Framework: CRAFT

To mitigate this, we propose **CRAFT** (Consensus Reasoning knowledge graph Aggregation for Flaw-aware Trace synthesis), a unified framework that mitigates both *Step Internal Flaws* (logical

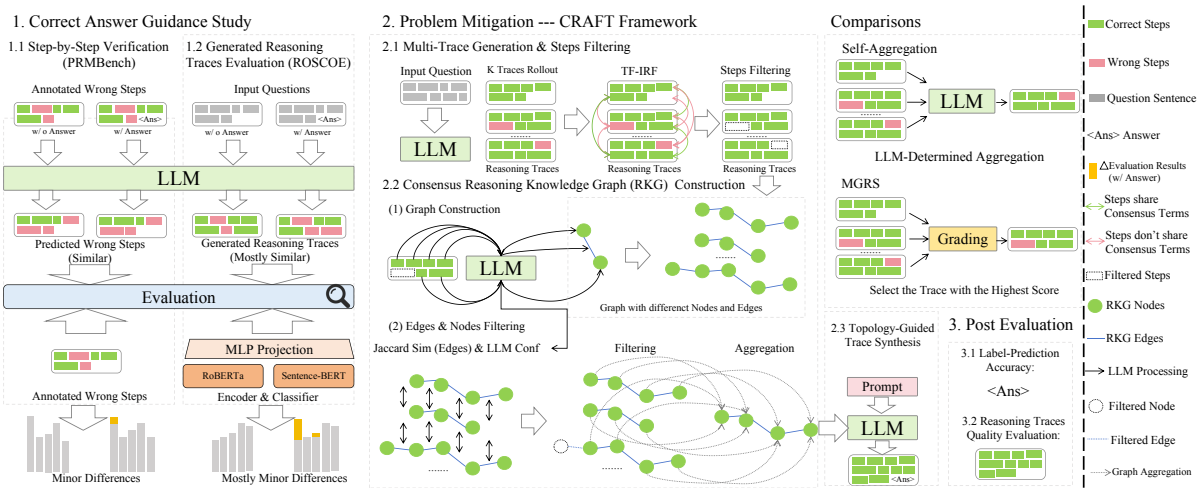

Figure 2: Overview of our work. **First**, we show that providing the correct answer does not consistently improve LLM reasoning ability, which motivates a structural approach for problem mitigation. **Second**, we show how CRAFT operates within the framework. **Third**, we test label-prediction accuracy on both logical and mathematical reasoning datasets, and post-processed reasoning traces on fine-grained benchmark evaluation.

errors, hallucinations, etc.) and *Step-wise Flaws* (overthinking, underthinking). CRAFT rolls out K diverse candidate traces per sample and leverages cross-trace consensus to detect both flaw types: steps with erroneous content fail to appear across other traces (Step Internal Flaws), while missing or redundant steps are revealed by comparing each trace's structure against others (Step-wise Flaws). Inspired by GRPO's group relative comparison (Guo et al., 2025), we identify flawed steps as deviations from intra-group consensus.

Inspired by Knowledge Graphs (Besta et al., 2024), we construct a *Reasoning Knowledge Graph* (RKG) in which nodes are reasoning steps and edges are logical relations, and RKGs G_S are aggregated into a consensus RKG G^* whose topology guides trace synthesis. CRAFT proceeds in three modules (Algorithm 1): **Module I** (Multi-Trace Generation & Steps Filtering) rolls out K candidate traces and extracts consensus terms T_{Con} via TF-IRF, and removes flawed steps via z-score filtering. **Module II** (Consensus RKG Construction) constructs a reasoning knowledge graph (RKG) for each reasoning trace, applies edge & node filtering, and aggregates them into a consensus RKG G^*; **Module III** (Topology-guided Trace Synthesis) generates one high-quality trace by traversing G^* in topological order.

Module I: Multi-Trace Generation & Steps Filtering For each sample, we roll out K candidate traces at temperature T. We then score each term by *TF-IRF* (Term Frequency–Inverse Reasoning Frequency), a variant of TF-IDF (Das et al., 2023) adapted to the multi-trace setting: terms frequent within one sample's K traces but rare across samples are considered informative. Crucially, TF is computed within K traces of one sample, and IRF is computed over all dataset samples (Appendix E, Eqs. 1–3). After filtering common logical connectives via a COMMONLOGICALWORDS blocklist (Bird et al., 2009), the Consensus Terms Set T_{Con} is defined directly by combining a per-term importance threshold and a cross-trace frequency threshold: $T_{\text{Con}} = \{w \mid \text{TF-IRF}(w) > \alpha \ \wedge \ \frac{1}{|K|}\sum_K \overline{\text{TF}}(w) \geq \beta\}$, where α is the TF-IRF importance threshold and β is the frequency threshold (fraction of traces in which a term must appear), and T_{Con} guides steps filtering.

For Steps Filtering, following GRPO's group-relative comparison (Guo et al., 2025), each step s is scored by its terms' Jaccard similarity with T_{Con}, z-normalized within the traces. Steps whose z-score falls below a cutoff γ (default -1.0, i.e., more than one standard deviation below the group mean) are removed as outliers, catching *Step Internal Flaws* (erroneous steps introduce terms absent from other $K-1$ traces) and *Step-wise Flaws* (redundant overthinking steps carry unusual terms).

Module II: Consensus RKG Construction This module constructs the consensus Reasoning Knowledge Graph (RKG) based on two parts:

(1) Graph Construction. After steps filtering, each trace is converted into an RKG G_S, via an LLM prompt that converts steps as nodes and logical relationships as edges.

(2) Edges & Nodes Filtering. Edge weight $\text{W}(e)$ is computed as an equal-weighted sum of the LLM-reported confidence score and the Jaccard Similarity between each two consecutive steps. Edges whose weight is below θ (the edge filtering

Algorithm 1: CRAFT for Reasoning Trace Improvement (One Sample)

```
// Module I: Multi-Trace Generation & Steps Filtering
for i = 1 to K do
    C ← C ∪ {LLM(Sample)};
T_Con ← { w | TF-IRF(w) > α ∧ (1/|K|) Σ_K TF(w) ≥ β };        // step-level consensus terms
foreach step s of trace S ∈ C do
    Z(s) ← (Jaccard(s, T_Con) − μ_S)/σ_S;        // Jaccard similarity, z-score steps filtering
    if Z(s) < γ then remove s from S;

// Module II: Consensus RKG Construction
// (1) Consensus RKG Construction
foreach trace S ∈ C do
    G_S = (V_S, E_S) ← LLM(S);        // Build RKG, steps→nodes, logical relations→edges
// (2) Edges & Nodes Filtering
foreach e=(u, v) ∈ ⋃_{S∈C} E_S do
    W(e) ← (1−λ) · conf(e) + λ · Jaccard(u, v);        // edge weight: LLM conf + node Jaccard
    G_S ← G_S \ {e : W(e)≤θ} \ {v : d⁺(v)=0 ∧ d⁻(v)=0};        // edge and node filtering
G* = ⋃_{S∈C} G_S ;        // aggregation into one graph
// Module III: Topology-guided Trace Synthesis
π ← (v_1, ..., v_|V*|) s.t. (v_i, v_j) ∈ E* ⇒ i < j;        // topological sort over G*
T_Final ← [] ;
foreach node v in π do
    s(v) ~ p_θ( · | Sample, {s(u) : (u, v) ∈ E*}, v );        // step-by-step generation
    Append s(v) to T_Final;
return T_Final;
```

threshold) are filtered, ensuring the graph retains relations supported by multiple candidate traces. Following this, we apply node filtering to G_S, removing isolated nodes whose in-degree and out-degree both equal zero, i.e., $d^{+}(v){=}0 \wedge d^{-}(v){=}0$. Per-trace graphs G_S are aggregated into one consensus RKG G^* by taking the set union. During Graph Construction in (1), nodes and edges produced by the LLM are identical when they correspond to identical steps and logical relationships; therefore, set aggregation of graphs introduces no redundant or repetitive elements. Detailed filtering statistics are in Tables 8 and 9.

Domain-specific Optimization. For logical reasoning, steps expressing the same inference with different words receive high LLM confidence, down-weighting Jaccard influence. For mathematical reasoning, the LLM-confidence component of $\mathrm{W}(e)$ further handles cases where equivalent expressions appear in different forms, and provably incorrect calculations are removed.

Module III: Topology-guided Trace Synthesis Since G^* is built via filtering and aggregation, edge directions are eliminated. We therefore perform a topological sort of G^* and generate the final reasoning trace step by step. Each step's prompt includes the corresponding node from the consensus RKG. Prompt templates are in Appendix M.

2.3 Mitigation of Different Flaw Types

The principle of flaw mitigation is a simple mechanism: steps shared by many candidate traces are preserved, while those shared by fewer are filtered.

Step Internal Flaws (logical errors, hallucinations, etc.). Error terms are absent from most candidate traces, so their steps score below γ under Module II's z-score filtering and are filtered.

Step-wise Flaws. *Underthinking*: steps missing from one trace are preserved by others in G^*. *Overthinking*: redundant steps carry unusual terms (low Z) and participate in low-weight edges (low $\mathrm{W}(e)$), so they can be removed by both z-score filtering and edge filtering in Module II.

3 Experiments

3.1 Experimental Setup of Correct Answer Guidance Study

We first validate the hypothesis: does providing correct answers to questions improve LLMs' rea-

Model	Settings	PRMBench				ROSCOE				
		Dataset	StepAcc	1stErr	F1	Dataset	Faithfulness	Info (Step)	Info (Chain)	Grammar
GPT-o4-mini	w/ Answer	Simplicity	0.79	0.64	0.87	CosmosQA	0.81	0.79	0.92	0.96
		Soundness	0.79	0.76	0.86	DROP	0.83	0.80	0.93	0.94
		Sensitivity	0.75	0.62	0.83	eSNLI	0.73	0.78	0.87	0.84
		Total	0.78	0.67	0.86	GSM8K	0.83	0.84	0.95	0.93
	w/o Answer	Simplicity	0.76	0.65	0.85	CosmosQA	0.81	0.80	0.92	0.93
		Soundness	0.79	0.74	0.87	DROP	0.83	0.81	0.93	0.95
		Sensitivity	0.74	0.60	0.83	eSNLI	0.71	0.78	0.87	0.83
		Total	0.76	0.66	0.85	GSM8K	0.84	0.85	0.96	0.93
Gemini-3-Flash (Thinking)	w/ Answer	Simplicity	0.76	0.70	0.85	CosmosQA	0.79	0.77	0.91	0.93
		Soundness	0.82	0.72	0.89	DROP	0.82	0.79	0.93	0.94
		Sensitivity	0.74	0.58	0.83	eSNLI	0.71	0.75	0.87	0.90
		Total	0.77	0.67	0.86	GSM8K	0.82	0.82	0.94	0.90
	w/o Answer	Simplicity	0.73	0.75	0.83	CosmosQA	0.79	0.78	0.91	0.95
		Soundness	**0.83**	0.70	**0.89**	DROP	0.83	0.81	0.92	**0.97**
		Sensitivity	0.70	0.58	0.80	eSNLI	0.68	0.75	0.87	0.88
		Total	0.75	0.68	0.84	GSM8K	0.82	0.84	0.96	0.94
GPT-5.4-nano	w/ Answer	Simplicity	0.62	0.63	0.74	CosmosQA	0.82	0.79	0.91	0.87
		Soundness	0.62	**0.79**	0.72	DROP	0.84	0.80	0.92	0.92
		Sensitivity	0.62	0.51	0.73	eSNLI	0.74	0.80	0.86	0.83
		Total	0.62	0.64	0.73	GSM8K	0.83	0.84	0.95	0.95
	w/o Answer	Simplicity	0.60	0.63	0.73	CosmosQA	0.84	0.81	0.92	0.88
		Soundness	0.69	0.71	0.79	DROP	0.83	0.80	0.92	0.94
		Sensitivity	0.62	0.51	0.74	eSNLI	0.68	0.77	0.86	0.84
		Total	0.64	0.62	0.75	GSM8K	0.81	0.83	0.96	0.92
DeepSeek-R1	w/ Answer	Simplicity	0.75	0.71	0.84	CosmosQA	0.79	0.78	0.91	0.94
		Soundness	0.79	0.71	0.86	DROP	0.83	0.81	0.93	0.96
		Sensitivity	0.71	0.50	0.80	eSNLI	0.72	0.78	0.88	0.88
		Total	0.75	0.64	0.84	GSM8K	0.83	0.85	0.95	0.93
	w/o Answer	Simplicity	0.74	0.63	0.84	CosmosQA	0.81	0.79	0.91	0.94
		Soundness	0.76	0.78	0.84	DROP	0.83	0.80	0.93	0.95
		Sensitivity	0.69	0.53	0.79	eSNLI	0.68	0.77	0.86	0.84
		Total	0.73	0.65	0.82	GSM8K	**0.85**	**0.85**	**0.96**	0.95

Table 1: Correct Answer Guidance Study: **PRMBench** step-level verifier (*Benchmark* = PRMBench error category; StepAcc, 1stErr, F1) and **ROSCOE** trace quality (four metrics applicable to all datasets).

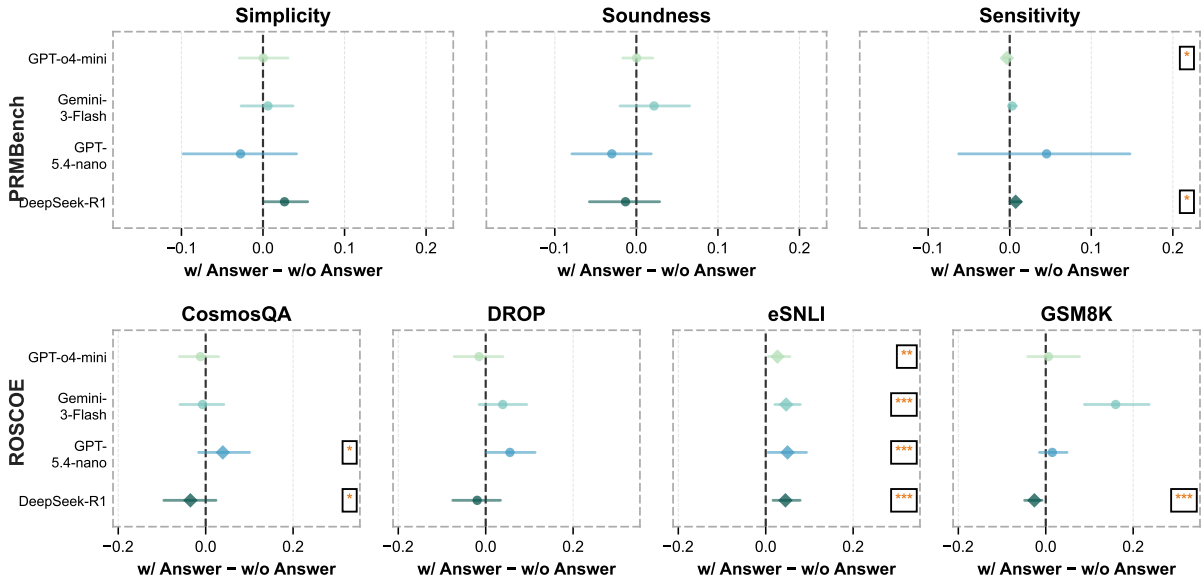

Figure 3: Significance testing (paired Wilcoxon, **w/ Answer** and **w/o Answer** settings, 95% bootstrap CI) on PRMBench (top) and ROSCOE (bottom). $*$: $p < 0.05$, $**$: $p < 0.01$, $***$: $p < 0.001$.

soning ability? We evaluate this using two benchmarks: (1) **PRMBench** (Song et al., 2025) benchmarks an LLM as a step-by-step verifier to detect erroneous steps across three datasets, each targeting one error type: *Simplicity* (redundant steps), *Soundness* (logically invalid steps), and *Sensitivity* (factually incorrect steps). Metrics include step accuracy (StepAcc), F1, and first-error identification accuracy (1stErr). (2) **ROSCOE** (Golovneva et al., 2023) evaluates the quality of reasoning traces

generated by the tested LLM. We test this on four datasets: CosmosQA, DROP, eSNLI, GSM8K, measuring *Faithfulness*, step-level and chain-level *Informativeness*, and *Grammar*.

We evaluate four backbone LLMs: GPT-o4-mini, Gemini-3-Flash (Thinking) (Comanici et al., 2025), GPT-5.4-nano, and DeepSeek-R1 (Guo et al., 2025). All models are set to temperature 0.7 with the same prompt templates (Appendix N.6), under two experimental settings: (1) **w/ Answer**: the correct answer is provided in the prompt, and (2) **w/o Answer**: no correct answer is provided.

3.2 Experimental Results

PRMBench. As shown in Table 1 and Figure 3, providing the correct answer yields no consistent improvement across three PRMBench datasets. Paired Wilcoxon signed-rank tests (metrics pooled per dimension; see Appendix L for details) confirm this finding: Simplicity and Soundness show *no significant difference* across the four models (mostly $p > 0.05$). Only Sensitivity reaches marginal significance for two models: o4-mini (p=0.028, negatively correlated) and DeepSeek-R1 (p=0.012, positively correlated). However, the opposing signs indicate no consistent benefit from providing the correct answer. In several cases, providing correct answers even hurts performance (e.g., GPT-5.4-nano Soundness F1 drops from 0.79 to 0.72). Detailed results are in Appendix A.

ROSCOE. DROP shows no significant influence on any model (all $p > 0.05$). The most consistent finding is eSNLI, where the **w/ answer** setting significantly improves quality for all four models ($p < 0.01$; $d = 0.23$–0.31), reflecting that NLI labels directly provide genuine help, which is absent in other tasks. Regarding this phenomenon, we discover that in the **w/o answer** setting, LLMs are strongly biased towards outputting the label "Neutral". The **w/ answer** setting offers the correct answer, which helps reshape the reasoning. Sporadic significance appears in CosmosQA (GPT-5.4-nano, DeepSeek-R1; $p < 0.05$) and GSM8K (DeepSeek-R1; $p < 0.001$, negatively correlated), but these effects are inconsistent across models. Detailed results are in Appendix A.

Takeaways. We draw three key findings:

(1) **In general, providing the correct answer does not improve reasoning ability.** Trace quality improves consistently only on eSNLI, and not across most other reasoning settings.

(2) **LLM reasoning flaws cannot be resolved by simply providing the correct answer.** Problems arise from the model's internal reasoning, not from uncertainty about the correct answer.

(3) **A unified mitigation framework is necessary**, as different samples exhibit different flaw types. This motivates CRAFT below.

3.3 Experimental Setup of CRAFT

As we have shown that providing the correct answer cannot improve LLMs' reasoning, we evaluate whether CRAFT's structural consensus approach succeeds. We measure label-prediction accuracy on four datasets. **Logical reasoning:** (1) FLD (Morishita et al., 2024) requires step-by-step deductive inference over natural language facts, (2) FOLIO (Han et al., 2024) tests first-order logic reasoning over everyday topics. **Mathematical reasoning:** (1) GSM8K (Cobbe et al., 2021) consists of grade-school math problems requiring multi-step reasoning; (2) OlympiadBench (He et al., 2024) contains competition-level mathematics problems demanding advanced reasoning. We select 500 samples for each logical reasoning dataset (250 labeled "Proved" and 250 "Disproved"), and 500 samples for each mathematical reasoning dataset, with random seed = 42. For evaluation, we use Exact-Match (EM) to get predicted labels; if it fails, we apply LLM-As-A-Judge for identification from reasoning traces. We manually verified 200 cases to ensure the reliability of evaluation.

We evaluate under two backbone models: **GPT-5.4-nano** (a lightweight reasoning model) and **o4-mini** (a reasoning model). Each model is used throughout the framework at 0.7 temperature.

3.4 Experimental Results

Table 2 compares CRAFT against thirteen baselines, details are in Appendix K.

CRAFT achieves the best or second-best accuracy in label-prediction, exceeding most baselines across logical reasoning and mathematical reasoning datasets. Gains are notable on harder datasets: on OlympiadBench (He et al., 2024), CRAFT outperforms the next-best baseline by up to 8.5% in accuracy while using over 70% fewer steps, simultaneously achieving the highest accuracy and the lowest step count. Notably, the reduction in step count comes without any accuracy trade-off: CRAFT uses roughly half the steps of most baselines, because Module II removes both *Step Inter-*

Setting	FLD			FOLIO			GSM8K		OlympiadBench	
	Acc(%)↑	F1↑	Avg.Steps(n)↓	Acc(%)↑	F1↑	Avg.Steps(n)↓	Acc(%)↑	Avg.Steps(n)↓	Acc(%)↑	Avg.Steps(n)↓
GPT-5.4-nano										
Best-of-N (Stiennon et al., 2022)	56.0	0.48	13.9	86.0	0.86	9.3	94.8	19.4	70.0	29.0
Self-Consistency (Wang et al., 2023)	61.8	0.55	14.5	80.0	0.80	9.5	93.0	17.9	65.3	29.0
Univ. Self-Consistency (Chen et al., 2023)	61.4	0.59	14.6	70.0	0.70	9.7	92.0	17.6	58.4	29.2
Self-Refine (Madaan et al., 2023)	63.4	0.59	20.5	86.0	0.86	13.0	91.4	19.1	63.4	29.6
Self-Eval Beam Search (Xie et al., 2023)	51.8	0.49	**1.4**	78.0	0.77	**1.4**	79.0	**4.4**	25.4	10.9
Faithful CoT (Lyu et al., 2023)	56.8	0.52	27.3	68.6	0.68	20.0	90.2	7.0	51.0	25.9
Tree-of-Thought (Yao et al., 2023)	24.0	0.28	10.5	74.0	0.81	9.0	90.6	6.2	46.6	**7.6**
RAP (Hao et al., 2023)	61.4	0.62	12.3	78.6	0.85	7.5	95.8	8.9	55.6	10.2
ConCISE (Qiao et al., 2025)	69.4	0.69	10.2	83.2	0.83	8.3	94.4	7.2	63.2	19.6
LCoT2Tree (Jiang et al., 2025)	69.4	0.68	11.5	85.4	0.84	5.9	94.8	8.9	65.6	10.2
DICE (Li et al., 2025)	70.0	0.68	12.2	86.6	0.85	10.8	95.0	8.8	68.8	14.4
PNS-Optimization (Yu et al., 2025)	69.8	0.68	5.4	83.6	0.83	7.8	95.0	5.9	67.4	11.3
Self-Aggregation (Venkatraman et al., 2026)	68.4	0.67	17.1	83.2	0.84	11.2	93.0	11.3	56.0	26.0
CRAFT (Ours)	**71.6**	**0.71**	9.9	**89.6**	**0.90**	5.8	**96.0**	5.4	**73.8**	**7.6**
o4-mini										
Best-of-N (Stiennon et al., 2022)	44.0	0.43	4.7	82.8	0.84	8.5	98.0	5.8	64.0	6.5
Self-Consistency (Wang et al., 2023)	60.6	0.58	6.0	82.0	0.85	7.0	95.6	5.3	68.0	29.4
Univ. Self-Consistency (Chen et al., 2023)	56.2	0.65	5.8	82.0	0.85	8.0	**98.5**	5.7	57.6	29.1
Self-Refine (Madaan et al., 2023)	44.8	0.51	9.3	76.0	0.80	10.6	93.2	5.8	68.0	29.7
Self-Eval Beam Search (Xie et al., 2023)	62.0	0.70	**4.1**	76.0	0.86	**3.2**	37.8	**2.2**	40.0	27.8
Faithful CoT (Lyu et al., 2023)	48.0	0.52	8.9	86.2	0.86	16.52	93.2	4.4	60.8	42.2
Tree-of-Thought (Yao et al., 2023)	52.4	0.48	4.3	78.6	0.85	7.5	91.8	6.5	46.0	**3.0**
RAP (Hao et al., 2023)	70.4	0.72	6.5	80.2	0.83	7.8	95.0	5.9	66.5	18.5
ConCISE (Qiao et al., 2025)	66.4	0.65	10.2	82.2	0.81	5.3	93.4	6.2	63.6	16.6
LCoT2Tree (Jiang et al., 2025)	71.4	0.70	12.3	85.4	0.85	7.5	97.4	8.9	65.6	10.2
DICE (Li et al., 2025)	72.6	0.72	6.5	85.2	0.83	7.8	96.0	5.9	68.8	14.3
PNS-Optimization (Yu et al., 2025)	71.6	0.72	13.8	85.4	0.84	9.3	94.6	5.6	72.2	10.2
Self-Aggregation (Venkatraman et al., 2026)	46.8	0.45	7.0	80.0	0.87	9.5	96.6	8.2	62.4	28.3
CRAFT (Ours)	**75.6**	**0.73**	7.2	**88.8**	**0.89**	7.8	98.0	5.5	**73.2**	6.6

Table 2: Benchmark results for two backbone models. Metrics: *Acc* (%), Macro-*F1*, and *Average Steps*. **Bold** = best; underline = second-best within each setting. Higher is better for *Acc* and *F1*; lower is better for *Steps*. F1 is omitted for mathematical reasoning datasets because each problem has a unique answer, making F1 = Accuracy.

Model	Dataset	Grammar↑	Rep-Step↓	Rep-Word↓
GPT-5.4-nano	CosmosQA	+1.6%	−1.7%	−1.7%
	DROP	+1.4%	−1.8%	−2.1%
	eSNLI	+2.4%	−1.5%	−1.2%
	GSM8k	+3.8%	−2.7%	−2.2%
o4-mini	CosmosQA	+2.1%	−2.0%	−2.8%
	DROP	+2.8%	−2.7%	−2.2%
	eSNLI	+5.9%	−2.0%	−1.4%
	GSM8k	+1.2%	−2.4%	−1.9%

Table 3: **ROSCOE** evaluation comparing the quality of CRAFT's post-processed traces against raw reasoning.

nal Flaws and *Step-wise Flaws*, while Module III synthesizes one high-quality reasoning trace.

We summarize the key findings as takeaways:

▷ **Consensus-driven synthesis consistently outperforms baselines.** All baselines rely on training-free methods; CRAFT synthesizes one trace based on the consensus part of candidate traces, yielding higher accuracy gains.

▷ **Step efficiency is a byproduct of accuracy gain.** CRAFT achieves the best accuracy while using far fewer steps.

▷ **Accuracy gains of CRAFT are generalizable.** Improvements are consistent across both logical and mathematical reasoning datasets.

▷ **LLMs get the majority of steps correct, even on hard problems.** While accuracy varies across datasets, accuracy gains of CRAFT generalize across both hard and easy datasets, implying that the majority of steps are correct. Otherwise, consensus aggregation across K traces would amplify errors, leading to accuracy drops.

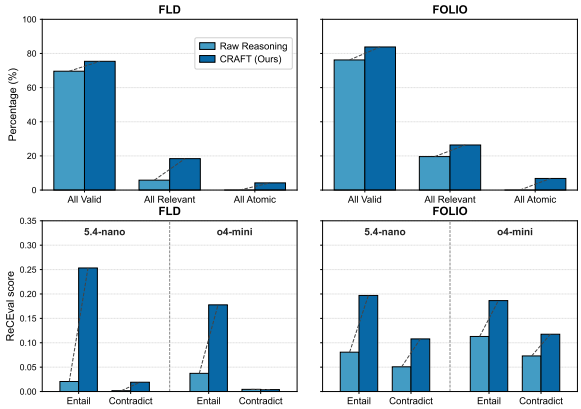

Figure 4: **Finelogic** (above two) and **ReCEval** (below two) evaluation comparing the quality of CRAFT's post-processed traces against raw reasoning.

3.5 Quality Evaluation of Post-Processed Reasoning Traces

We evaluate whether CRAFT improves reasoning trace quality by running ROSCOE (Golovneva et al., 2023) and ReCEval (Prasad et al., 2023) on GPT-5.4-nano and o4-mini, and FineLogic (Zhou et al., 2025) on GPT-5.4-nano.

Results of ROSCOE are in Table 3, which reports metrics measuring trace quality: (1) *Grammar* (grammar correctness per step), (2) *Rep-Step* (step-level redundancy), and (3) *Rep-Word* (word-level redundancy). Figure 4 reports the results of FineLogic evaluation: (i) *All Valid*: all steps are logically valid; (ii) *All Relevant*: all steps contribute to the conclusion; (iii) *All Atomic*: all steps

Ablation Setting	GPT-5.4-nano		o4-mini	
	Acc(%)	Δ(%)	Acc(%)	Δ(%)
w/o CRAFT	47.8	−26.0	48.0	−25.2
w/o RKG	67.6	−6.2	56.0	−17.2
w/o Synthesis	52.0	−21.8	57.8	−15.4
w/o Filter & Synthesis	49.0	−24.8	54.0	−19.2
w/o Weighted Edges Fusion	60.2	−13.6	57.2	−16.6
Embedding Cosine Similarity (Replacing Jaccard Similarity)	71.8	−2.0	71.4	−1.8

Table 4: Ablation study of CRAFT on OlympiadBench (73.8% / 73.2%). Δ(%) = Accuracy drop.

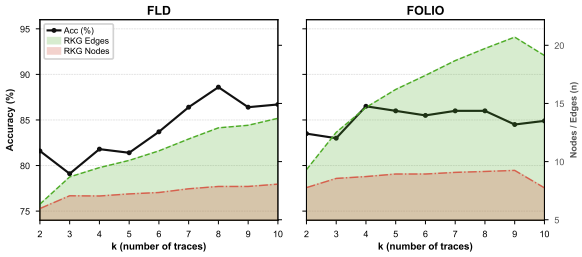

Figure 5: Accuracy increases then converge, with the number of traces K increases on GPT-5.4-nano.

convey an atomic relationship. It also reports the results of ReCEval: (1) *Entail*: whether this step can be inferred from previous steps; (2) *Contradict*: whether this step is NOT contradictory to previous steps. Higher is better for all metrics.

4 Ablation Study

Table 4 validates the effectiveness of each component by removing them on OlympiadBench and reporting accuracy drops compared to full CRAFT.

Ablating the whole CRAFT pipeline leads to accuracy drops of 20% or more. Removing RKG (−6.2% / −17.2%) or removing synthesis (−21.8% / −15.4%) account for large drops, and removing Filtering (w/o Filtering & Synthesis) yields further drops. Directly applying LLM-constructed RKG edges (w/o Weighted Edges Fusion) results in −13.6%/ −16.6% accuracy drop. Replacing Jaccard with `text-embedding-3-large` cosine similarity costs about −2.0% / −1.8% accuracy drop, confirming Jaccard is a better choice: maintaining higher accuracy without additional API call.

5 Hyperparameter Settings

Sensitivity of K (number of traces). As shown in Figure 5, label-prediction accuracy varies with K from 2 to 10. For FLD, accuracy increases gradually from K=3 onward. For FOLIO, peak accuracy is reached earlier, at about $K = 3$. As the number of traces K increases, the growth rate of RKG edges consistently outpaces that of nodes, reaching a higher value before slowing

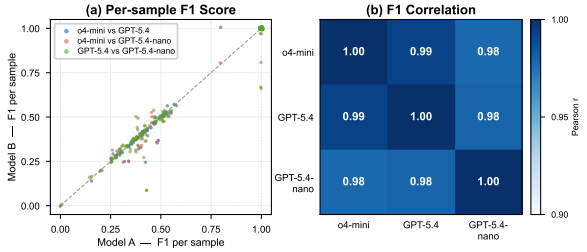

Figure 6: Graph Construction G_S edge extraction F1, points cluster around the diagonal (left), and Pearson r of pair-wise models' results (right).

down. Moreover, the size of RKGs is positively correlated with accuracy. Detailed setup is in Appendix G; computational costs are in Appendix K.

Other hyperparameters like TF-IRF threshold β=0.3, z-score steps filtering threshold γ=−1.0, edge weight parameter λ=0.5, and edge weight filtering threshold θ=0.3, were set once and not changed, reducing overfitting risks.

6 Noise Analysis on Graph Construction

Graph construction G_S relies on an LLM prompt to identify step dependencies. To verify the robustness, we evaluate edge extraction F1 against ground-truth annotations on FLD reasoning traces, across three backbone models (GPT-5.4-nano, o4-mini, GPT-5.4) under identical prompts and 0.7 temperature. F1 varies by less than 2% (0.57–0.59) across model tiers, with pair-wise Pearson $r \geq 0.98$ per sample (Figure 6), confirming that the quality of graph construction does not introduce explicit model-specific noise.

7 Conclusion

In this work, we show that providing correct answers does not consistently improve LLMs' reasoning ability, which clarifies the problem: reasoning flaws lie in the structure of generated traces, rather than in the exposure to the correct answer. We categorize reasoning flaws into *Step Internal Flaws* and *Step-wise Flaws*, and propose CRAFT, a unified framework that mitigates both flaw types through cross-trace consensus. CRAFT builds a consensus Reasoning Knowledge Graph (RKG), then synthesizes a high-quality trace via topological generation. CRAFT consistently outperforms baselines in label-prediction accuracy across logical and mathematical reasoning datasets. Post-processed reasoning traces achieve higher quality on fine-grained benchmark evaluation. Our work suggests that applying trace-wise consensus can be a promising direction for future LLM reasoning development.

Limitations

Our work has the following limitations:

(1) **Sequential CoT assumption.** CRAFT is designed for sequential Chain-of-Thought traces and may fall short on parallel or tree-structured reasoning paradigms.

(2) **Consensus $\neq$ correctness.** The framework assumes that steps and edges appearing frequently across K traces are more likely correct. On hard problems where many candidates share the same wrong reasoning pattern (e.g., a common post-hoc rationalisation), consensus can amplify rather than suppress errors. Increasing K mitigates this (Section 5), but does not eliminate it.

(3) **TF-IRF on small K.** IRF is computed over only K traces per sample rather than a large corpus; with small K the frequency signal may lack discriminative power, making term importance estimates noisier.

Ethical Considerations

Our work focuses on both logical and mathematical reasoning domains. We do not address sensitive topics like race or gender. All evaluations are carried out on established benchmarks. We release all our code and datasets at `https://anonymous.4open.science/r/CRAFT-2712/`.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Sebastopol, CA.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2023. Universal self-consistency for large language model generation. *Preprint*, arXiv:2311.17311.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3416 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Mamata Das, Selvakumar K., and P. J. A. Alphonse. 2023. A comparative study on tf-idf feature weighting method and its analysis using unstructured dataset. *Preprint*, arXiv:2308.04037.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*.

Bryan Eikema and Wilker Aziz. 2020. Is map decoding all you need? the inadequacy of the mode in neural machine translation. *Preprint*, arXiv:2005.10283.

Olga Golovneva, Moya Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Roscoe: A suite of metrics for scoring step-by-step reasoning. *Preprint*, arXiv:2212.07919.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, and 16 others. 2024. Folio: Natural language reasoning with first-order logic. *Preprint*, arXiv:2209.00840.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *Preprint*, arXiv:2305.14992.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *Preprint*, arXiv:2402.14008.

Gangwei Jiang, Yahui Liu, Zhaoyi Li, Qi Wang, Fuzheng Zhang, Linqi Song, Ying Wei, and Defu Lian. 2025. What makes a good reasoning chain?

uncovering structural patterns in long chain-of-thought reasoning. *Preprint*, arXiv:2505.22148.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. *Preprint*, arXiv:2404.07103.

Yiqi Li, Yusheng Liao, Zhe Chen, Yanfeng Wang, and Yu Wang. 2025. Dice: Structured reasoning in llms through slm-guided chain-of-thought correction. *Preprint*, arXiv:2510.09211.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Zipeng Ling, Yuehao Tang, Chen Huang, Shuliang Liu, Gaoyang Jiang, Shenghong Fu, Junqi Yang, Yao Wan, Jiawan Zhang, Kejia Huang, and Xuming Hu. 2025a. Instruction boundary: Quantifying biases in llm reasoning under various coverage. *Preprint*, arXiv:2509.20278.

Zipeng Ling, Yuehao Tang, Shuliang Liu, Junqi Yang, Shenghong Fu, Chen Huang, Kejia Huang, Yao Wan, Zhichao Hou, and Xuming Hu. 2025b. Wakenllm: Evaluating reasoning potential and stability in llms via fine-grained benchmarking. *Preprint*, arXiv:2507.16199.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. *Preprint*, arXiv:2301.13379.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *Preprint*, arXiv:2303.17651.

Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. 2024. Enhancing reasoning capabilities of llms via principled synthetic logic corpus. *Preprint*, arXiv:2411.12498.

Archiki Prasad, Swarnadeep Saha, Xiang Zhou, and Mohit Bansal. 2023. Receval: Evaluating reasoning chains via correctness and informativeness. *Preprint*, arXiv:2304.10703.

Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Guanbo Wang, Fandong Meng, Jie Zhou, Ju Ren, and Yaoxue Zhang. 2025. ConCISE: Confidence-guided compression in step-by-step efficient reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8010–8029, Suzhou, China. Association for Computational Linguistics.

Thomas Savage, Ashwin Nayak, Robert Gallo, Ekanath Rangan, and Jonathan H Chen. 2023. Diagnostic reasoning prompts reveal the potential for large language model interpretability in medicine. *Preprint*, arXiv:2308.06834.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. Prmbench: A fine-grained and challenging benchmark for process-level reward models. *Preprint*, arXiv:2501.03124.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. 2022. Learning to summarize from human feedback. *Preprint*, arXiv:2009.01325.

Siddarth Venkatraman, Vineet Jain, Sarthak Mittal, Vedant Shah, Johan Obando-Ceron, Yoshua Bengio, Brian R. Bartoldson, Bhavya Kailkhura, Guillaume Lajoie, Glen Berseth, Nikolay Malkin, and Moksh Jain. 2026. Recursive self-aggregation unlocks deep thinking in large language models. *Preprint*, arXiv:2509.26626.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. 2025. Evaluating mathematical reasoning beyond accuracy. *Preprint*, arXiv:2404.05692.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. 2023. Self-evaluation guided beam search for reasoning. *Preprint*, arXiv:2305.00633.

Fangzhi Xu, Qika Lin, Jiawei Han, Tianzhe Zhao, Jun Liu, and Erik Cambria. 2024. Are large language models really good logical reasoners? a comprehensive evaluation and beyond. *Preprint*, arXiv:2306.09841.

Haolei Xu, Yuchen Yan, Yongliang Shen, Wenqi Zhang, Guiyang Hou, Shengpei Jiang, Kaitao Song, Weiming Lu, Jun Xiao, and Yueting Zhuang. 2025. Mind the gap: Bridging thought leap for improved chain-of-thought tuning. *Preprint*, arXiv:2505.14684.

Jundong Xu, Hao Fei, Huichi Zhou, Xin Quan, Qijun Huang, Shengqiong Wu, William Yang Wang, Mong-Li Lee, and Wynne Hsu. 2026. Logicreward: Incentivizing llm reasoning via step-wise logical supervision. *Preprint*, arXiv:2512.18196.

Wei Yang, Shixuan Li, Heng Ping, Peiyu Zhang, Paul Bogdan, and Jesse Thomason. 2026. Auditing multi-agent llm reasoning trees outperforms majority vote and llm-as-judge. *arXiv preprint arXiv:2602.09341*.

Yujiao Yang, Jing Lian, and Linhui Li. 2025. Multi-chain graph refinement and selection for reliable reasoning in large language models. *arXiv preprint arXiv:2511.23136*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Fangyi Yu, Lee Quartey, and Frank Schilder. 2022. Legal prompting: Teaching a language model to think like a lawyer. *Preprint*, arXiv:2212.01326.

Xiangning Yu, Zhuohan Wang, Linyi Yang, Haoxuan Li, Anjie Liu, Xiao Xue, Jun Wang, and Mengyue Yang. 2025. Causal sufficiency and necessity improves chain-of-thought reasoning. *Preprint*, arXiv:2506.09853.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *Preprint*, arXiv:2504.13837.

Yilun Zhao, Hongjun Liu, Yitao Long, Rui Zhang, Chen Zhao, and Arman Cohan. 2024. Financemath: Knowledge-intensive math reasoning in finance domains. *Preprint*, arXiv:2311.09797.

Yujun Zhou, Jiayi Ye, Zipeng Ling, Yufei Han, Yue Huang, Haomin Zhuang, Zhenwen Liang, Kehan Guo, Taicheng Guo, Xiangqi Wang, and Xiangliang Zhang. 2025. Dissecting logical reasoning in llms: A fine-grained evaluation and supervision study. *Preprint*, arXiv:2506.04810.

A Full Benchmark Evaluation Results

Tables 5 and 6 report the complete PRMBench and ROSCOE results with all metrics.

B LLM Usage Statement

GPT-5.2 was used to help with grammar polishing. Claude-4.6-Sonnet was used to assist with code writing and organization. For the unsupervised comparison part in our methodology, we consulted Claude-4.6-Sonnet to select the best idea (manually created but polished up by AI, we then iteratively optimize this) to meet our requirements.

C Related Work

LLM Reasoning Flaws. Chain-of-Thought prompting (Wei et al., 2023) makes intermediate steps explicit, but generated traces suffer from two flaw categories: *Step Internal Flaws* (logical errors (Xu et al., 2024), hallucinations (Xia et al., 2025), etc.) and *Step-wise Flaws* (underthinking (Xu et al., 2025) and overthinking (Yu et al., 2025)). The single-flaw assumption cannot describe practical scenarios, where different samples exhibit different flaw types.

Reasoning Trace Improvement. Recursive Self-Aggregation (Venkatraman et al., 2026) iteratively combine subsets of intermediate steps across candidate chains, exploiting intra-chain structure, but without an explicit anomaly filtering. Multi-chain graph methods such as MGRS (Yang et al., 2025) construct a graph over multiple chains, but ultimately select the highest-scoring trace rather than making use of others. CRAFT applies cross-trace consensus and anomaly detection to mitigate different flaw types.

Multi-trace voting and selection. Self-Consistency (Wang et al., 2023) and Universal Self-Consistency (Chen et al., 2023) aggregate K candidate traces by majority vote over final answers, treating each trace as a black box and discarding all structural information within reasoning steps. Best-of-N (Stiennon et al., 2022) scores candidates with a reward model and returns the highest-ranked one. MBR-like (Minimum Bayes Risk) selection (Eikema and Aziz, 2020) chooses the candidate that minimises expected loss against other candidates. All three paradigms operate at the level of final answers or holistic scores; none inspects or improves the internal step structure. CRAFT's consensus RKG can be viewed as a structural generalisation of MBR: rather than selecting one existing candidate, it synthesises a new trace that captures the consensus structure across all K candidates.

Iterative self-refinement. Self-Refine (Madaan et al., 2023) iteratively revises a single trace by prompting the model to critique and rewrite its own output. Self-Evaluation Guided Beam Search (Xie et al., 2023) uses the model's self-assigned scores to guide beam search over partial traces. Both methods rely on the model's own judgment as the quality signal, which amplifies errors when the model is confidently wrong—a particularly acute problem for hard logical or mathematical reasoning. CRAFT instead derives its quality signal from cross-trace structural consensus, an external statistical measure that does not depend on the model's self-assessment.

Cross-chain synthesis. Recursive Self-Aggregation (RSA) (Venkatraman et al., 2026) iteratively draws subsets from a population of candidate chains, recombines their intermediate steps, and produces a refined population over multiple rounds. Unlike simple voting methods, RSA explicitly exploits the content of reasoning steps across chains. However, RSA lacks a structural consensus graph and an explicit anomaly-filtering phase: it recombines steps without first identifying which steps are structurally anomalous relative to the population. CRAFT's RKG provides both: edge weights and consensus node counts supply the statistical signal for filtering before synthesis.

Graph-of-Thought and single-trace graph methods. Graph-of-Thought (Besta et al., 2024) and GraphCoT (Jin et al., 2024) model the reasoning process as a directed graph within a single generation episode, enabling backtracking and refinement. Because the graph is built from one trace at a time, there is no cross-trace consensus signal: the method cannot detect steps that are anomalous relative to a population of candidate solutions. CRAFT's RKG aggregation across K traces provides edge-weight and consensus-node statistics that single-trace graph methods fundamentally lack.

Multi-chain graph refinement and selection. MGRS (Yang et al., 2025) constructs a directed acyclic graph over multiple candidate chains, estimates single-step and cumulative success rates at intermediate nodes via cross-chain verification,

Table 5: PRMBench step-verifier results across three error categories (full metrics). *w/ Answer/w/o Answer*: with/without correct answer. *StepAcc*: overall step accuracy; *CorrAcc*: accuracy on correct steps; *WrongAcc*: accuracy on erroneous steps; *1stErr*: first-error localization accuracy; *Prec/Rec/F1*: positive class; *neg-F1*: erroneous-step F1. Higher is better for all.

Model	Setting	Dim	StepAcc	CorrAcc	WrongAcc	1stErr	Prec	Recall	F1	neg-F1
GPT-o4-mini	w/ Answer	Simplicity	0.79	0.83	0.53	0.64	0.93	0.83	0.87	0.38
		Soundness	0.79	0.80	0.71	0.76	0.93	0.80	0.86	0.54
		Sensitivity	0.75	0.81	0.53	0.62	0.86	0.81	0.83	0.47
		Total	0.78	0.81	0.59	0.67	0.91	0.81	0.86	0.47
	w/o Answer	Simplicity	0.76	0.79	0.55	0.65	0.93	0.79	0.85	0.36
		Soundness	0.79	0.81	0.70	0.74	0.93	0.81	0.87	0.54
		Sensitivity	0.74	0.79	0.52	0.60	0.87	0.79	0.83	0.45
		Total	0.76	0.80	0.59	0.66	0.91	0.80	0.85	0.45
Gemini-3-Flash (Thinking)	w/ Answer	Simplicity	0.76	0.78	0.62	0.70	0.94	0.78	0.85	0.37
		Soundness	0.82	0.83	0.71	0.72	0.94	0.83	0.89	0.54
		Sensitivity	0.74	0.77	0.63	0.58	0.91	0.77	0.83	0.46
		Total	0.77	0.79	0.65	0.67	0.93	0.79	0.86	0.45
	w/o Answer	Simplicity	0.73	0.74	0.62	0.75	0.94	0.74	0.83	0.34
		Soundness	0.83	0.85	0.68	0.70	0.94	0.85	0.89	0.55
		Sensitivity	0.70	0.74	0.56	0.58	0.88	0.74	0.80	0.41
		Total	0.75	0.78	0.62	0.68	0.92	0.78	0.84	0.43
GPT-5.4-nano	w/ Answer	Simplicity	0.62	0.62	0.61	0.63	0.92	0.62	0.74	0.27
		Soundness	0.62	0.58	0.79	0.79	0.94	0.58	0.72	0.40
		Sensitivity	0.62	0.64	0.55	0.51	0.85	0.64	0.73	0.37
		Total	0.62	0.61	0.65	0.64	0.91	0.61	0.73	0.35
	w/o Answer	Simplicity	0.60	0.61	0.53	0.63	0.91	0.61	0.73	0.24
		Soundness	0.69	0.68	0.70	0.71	0.93	0.68	0.79	0.41
		Sensitivity	0.62	0.67	0.44	0.51	0.83	0.67	0.74	0.32
		Total	0.64	0.65	0.55	0.62	0.89	0.65	0.75	0.32
DeepSeek-R1	w/ Answer	Simplicity	0.75	0.77	0.56	0.71	0.93	0.77	0.84	0.35
		Soundness	0.79	0.81	0.73	0.71	0.93	0.81	0.86	0.57
		Sensitivity	0.71	0.74	0.55	0.50	0.87	0.74	0.80	0.43
		Total	0.75	0.77	0.62	0.64	0.91	0.77	0.84	0.45
	w/o Answer	Simplicity	0.74	0.77	0.56	0.63	0.93	0.77	0.84	0.35
		Soundness	0.76	0.76	0.74	0.78	0.93	0.76	0.84	0.53
		Sensitivity	0.69	0.72	0.56	0.53	0.87	0.72	0.79	0.42
		Total	0.73	0.75	0.62	0.65	0.91	0.75	0.82	0.43

and selects the trajectory with the highest cumulative score. This cross-chain verification step goes beyond simple voting, but MGRS ultimately returns the highest-scoring *original* trace rather than synthesising a new one. CRAFT synthesises a novel trace by traversing the consensus RKG in topological order, guided by consensus node texts as reference anchors; the same structural signal is used for both anomaly detection and synthesis, which MGRS does not perform.

Step-level intervention methods. CoNCISE (Qiao et al., 2025) fine-tunes large reasoning models on confidence-guided data (via SimPO) to suppress two identified patterns of redundant reflection—*Confidence Deficit* and *Termination Delay*—yielding shorter traces. This is a supervised compression method targeting *Step-wise Flaws* (overthinking) only; it requires training data and does not address Step Internal Flaws. DICE (Li et al., 2025) uses a trained small language model to post-process and correct the output format of a large model's chain-of-thought, enforcing structural compliance rather than improving reasoning content. PNS-Optimization (Yu et al., 2025) curates training data by scoring each reasoning step with a causal Probability of Necessity and Sufficiency estimator, filtering out causally redundant steps before fine-tuning. All three require a supervised training stage and target a single flaw dimension; CRAFT is training-free and addresses both flaw categories via cross-trace structural consensus.

Table 6: ROSCOE reasoning trace quality metrics across four datasets and two settings (full metrics). *w/ Answer/w/o Answer*: with/without correct answer. *Faith.*=Faithfulness, *Info-S*=Informativeness (step), *Info-C*=Informativeness (chain), *R-Align*=Reasoning Alignment, *Ext-Hall*=External Hallucination, *Redund.*=Redundancy, *Missing*=Missing Step, *Cov-S/C*=Semantic Coverage (step/chain), *Gram.*=Grammar. Metrics marked * only available for eSNLI and GSM8K.

Model	Set.	Dataset	Faith.	Info-S	Info-C	R-Align*	Ext-Hall*	Redund.*	Missing*	Cov-S*	Cov-C*	Gram.
GPT-o4-mini	w/ Answer	CosmosQA	0.81	0.79	0.92	—	—	—	—	—	—	0.96
		DROP	0.83	0.80	0.93	—	—	—	—	—	—	0.94
		eSNLI	0.73	0.78	0.87	0.77	0.86	0.66	0.79	0.95	0.90	0.84
		GSM8K	0.83	0.84	0.95	0.86	0.94	0.78	0.65	0.94	0.96	0.93
	w/o Answer	CosmosQA	0.81	0.80	0.92	—	—	—	—	—	—	0.93
		DROP	0.83	0.81	0.93	—	—	—	—	—	—	0.95
		eSNLI	0.71	0.78	0.87	0.74	0.80	0.59	0.77	0.95	0.90	0.83
		GSM8K	0.84	0.85	0.96	0.86	0.94	0.78	0.69	0.94	0.97	0.93
Gemini-3-Flash (Thinking)	w/ Answer	CosmosQA	0.79	0.77	0.91	—	—	—	—	—	—	0.93
		DROP	0.82	0.79	0.93	—	—	—	—	—	—	0.94
		eSNLI	0.71	0.75	0.87	0.77	0.86	0.67	0.78	0.94	0.90	0.90
		GSM8K	0.82	0.82	0.94	0.83	0.94	0.77	0.60	0.94	0.94	0.90
	w/o Answer	CosmosQA	0.79	0.78	0.91	—	—	—	—	—	—	0.95
		DROP	0.83	0.81	0.92	—	—	—	—	—	—	0.97
		eSNLI	0.68	0.75	0.87	0.73	0.80	0.59	0.79	0.93	0.89	0.88
		GSM8K	0.82	0.84	0.96	0.86	0.93	0.77	0.76	0.96	0.97	0.94
GPT-5.4-nano	w/ Answer	CosmosQA	0.82	0.79	0.91	—	—	—	—	—	—	0.87
		DROP	0.84	0.80	0.92	—	—	—	—	—	—	0.92
		eSNLI	0.74	0.80	0.86	0.77	0.86	0.64	0.77	0.94	0.90	0.83
		GSM8K	0.83	0.84	0.95	0.86	0.94	0.79	0.65	0.94	0.96	0.95
	w/o Answer	CosmosQA	0.84	0.81	0.92	—	—	—	—	—	—	0.88
		DROP	0.83	0.80	0.92	—	—	—	—	—	—	0.94
		eSNLI	0.68	0.77	0.86	0.74	0.80	0.57	0.74	0.93	0.89	0.84
		GSM8K	0.81	0.83	0.96	0.81	0.92	0.76	0.70	0.92	0.95	0.92
DeepSeek-R1	w/ Answer	CosmosQA	0.79	0.78	0.91	—	—	—	—	—	—	0.94
		DROP	0.83	0.81	0.93	—	—	—	—	—	—	0.96
		eSNLI	0.72	0.78	0.88	0.79	0.86	0.64	0.79	0.94	0.90	0.88
		GSM8K	0.83	0.85	0.95	0.86	0.94	0.71	0.64	0.94	0.96	0.93
	w/o Answer	CosmosQA	0.81	0.79	0.91	—	—	—	—	—	—	0.94
		DROP	0.83	0.80	0.93	—	—	—	—	—	—	0.95
		eSNLI	0.68	0.77	0.86	0.73	0.80	0.58	0.79	0.93	0.89	0.84
		GSM8K	0.85	0.85	0.96	0.86	0.94	0.78	0.66	0.94	0.97	0.95

Process-reward verifier-based methods. Process-reward models (PRMs) (Lightman et al., 2023) train dedicated step-level verifiers on human annotations, operating on individual traces in isolation. AgentAuditor (Yang et al., 2026) audits multi-agent reasoning trees with an LLM judge. Both require a trained or prompted verifier model as an external component. CRAFT instead derives quality signals structurally from cross-trace consensus: flawed steps are identified because they deviate from the majority pattern across K independent rollouts, without any separate judge model or annotated training data. The two approaches are orthogonal and could be composed—for instance, a PRM could re-rank CRAFT's synthesis candidates.

Debate-style methods. Multi-agent debate (Du et al., 2024) lets multiple LLM instances propose and critique conclusions through iterative argumentation. CRAFT differs by constructing a structural consensus across K independently generated traces and synthesising a new trace in topological order, rather than debating among existing candidates. Debate operates at the level of claim exchange; CRAFT operates at the level of step-graph topology. The two paradigms could be combined: debate could serve as a post-synthesis verification step.

D Dataset Details

We use seven reasoning benchmarks for two parts of experiments. (1) PRMBench and ROSCOE are carried out as an empirical validation: we show that in most settings, offering the correct answer does not improve the reasoning ability of LLMs. (2) The latter five, including ROSCOE, are implemented to validate whether the proposed framework exceeds baselines, whether in label-prediction accuracy or reasoning trace quality.

Reasoning Quality Evaluation (Section 2.1)

PRMBench. PRMBench (Song et al., 2025) evaluates the ability of LLMs to act as process reward models (step verifiers) on mathematical reasoning traces. It covers nine error categories and reports step-level accuracy, first-error detection rate, and F1 score.

ROSCOE. ROSCOE (Golovneva et al., 2023) is a reference-free evaluation suite for reasoning chain quality, measuring faithfulness, informativeness, coherence, and logical consistency. We use it in Section 2.1 to test whether the correct final answer improves the intrinsic quality of generated reasoning traces across multiple dimensions.

Logical Reasoning

FLD. The Formal Logic Dataset (FLD) (Morishita et al., 2024) is a synthetic benchmark constructed from propositional and first-order logic templates. Each instance has an explicitly specified logical structure with binary labels (PROVED/DISPROVED), minimizing linguistic shortcuts so that model errors directly reflect failures in logical reasoning. We evaluate on 500 balanced samples (250 per label).

FOLIO. FOLIO (Han et al., 2024) is a human-authored first-order logic benchmark where natural language statements are annotated with formal FOL representations. It supports ternary labels (PROVED/DISPROVED/UNKNOWN) and preserves realistic linguistic diversity, making it harder than purely synthetic datasets. We evaluate on 500 balanced samples (250 PROVED/250 DISPROVED). UNKNOWN examples are excluded because the consensus-based framework is designed for binary entailment decisions; including an "undecidable" class would conflate framework failures with genuine undecidability, following the evaluation protocol of Han et al. (2024).

Mathematical Reasoning

GSM8K. GSM8K (Cobbe et al., 2021) is a grade-school math word problem benchmark requiring multi-step arithmetic reasoning. Each problem has a unique numerical answer; as a result, Precision = Recall = Accuracy, so we report accuracy only. We evaluate on 500 problems from the test split.

OlympiadBench. OlympiadBench (He et al., 2024) contains competition-level mathematics problems sourced from national and international olympiads, spanning algebra, combinatorics, geometry, and number theory. It requires deep multi-step reasoning significantly beyond grade-school level. As with GSM8K, each problem has a unique answer, so F1 equals accuracy and only accuracy is reported. We evaluate on 500 problems from the English subset.

FineLogic. FineLogic (Zhou et al., 2025) offers a fine-grained benchmark for evaluating the step-level logical quality of reasoning traces. Rather than assigning a single chain-level score, it decomposes trace quality into three binary criteria evaluated per step: *All Valid* (every step is logically valid with respect to the premises), *All Relevant* (every step contributes to deriving the conclusion), and *All Atomic* (every step expresses exactly one atomic logical relationship). A trace passes a criterion only if *all* steps satisfy it, making FineLogic strictly sensitive to local reasoning flaws.

E TF-IRF Formulas

For each term w (excluding COMMONLOGICALWORDS (Bird et al., 2009)), we compute three quantities across the K traces of a sample:

$$\overline{\mathrm{TF}}(w) = \frac{1}{K}\sum_{k=1}^{K} \frac{\#(w, t_k)}{\sum_{w'} \#(w', t_k)}, \quad (1)$$

$$\overline{\mathrm{RF}}(w) = \frac{1}{N}\sum_{n=1}^{N} \frac{|\{k : w \in t_k^{(n)}\}|}{K}, \quad (2)$$

$$\text{TF-IRF}(w) = \overline{\mathrm{TF}}(w) \cdot \log\Big(1 + \frac{N}{\overline{\mathrm{RF}}(w)+1}\Big), \quad (3)$$

where t_k denotes trace k, $\#(w, t_k)$ the count of w in t_k, and N is the total number of samples. The consensus term set is defined in a single step by combining the importance threshold α and the cross-trace frequency threshold β: $T_{\text{Con}} = \{w \mid \text{TF-IRF}(w) > \alpha \wedge \frac{1}{|K|}\sum_K \overline{\mathrm{TF}}(w) \geq \beta\}$.

F Related Concepts

F.1 Z-Score Filtering

To improve the robustness of reasoning trace evaluation, we adopt a Z-score based filtering strategy to detect abnormal responses within a group of generated traces. The intuition is that reasoning traces with significantly different metric values (e.g., logical consistency score, step completeness, or similarity score) may correspond to unstable or low-quality reasoning. Given a metric value x_i from a set of responses $\{x_1, x_2, \ldots, x_n\}$ gen-

erated under the same prompt, we compute the Z-score as:

$$Z_i = \frac{x_i - \mu}{\sigma},$$

where μ is the mean and σ is the standard deviation of the group. Responses whose absolute Z-score exceeds a predefined threshold (e.g., $|Z_i| > 2$) are treated as outliers and removed from subsequent aggregation or comparison. This filtering step reduces the influence of extreme reasoning traces and stabilizes downstream statistics such as group agreement and correctness estimation. In our implementation, Z-score filtering is applied at the response-group level before computing similarity-based agreement metrics.

F.2 Jaccard Similarity for Reasoning Trace Agreement

To quantify agreement among reasoning traces, we measure similarity at the token-set or step-set level using Jaccard similarity. This metric captures the overlap between reasoning components while remaining robust to variations in phrasing.

Given two reasoning traces represented as sets of elements (e.g., reasoning steps, keywords, or normalized tokens), A and B, the Jaccard similarity is defined as:

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

This score ranges from 0 to 1, where higher values indicate stronger agreement between traces.

In CRAFT, reasoning traces within the same prompt group are first normalized into structured step representations. Pairwise Jaccard similarity is then computed across traces, and group-level agreement is obtained by averaging pairwise similarities.

This similarity-based agreement measure provides a lightweight proxy for reasoning consistency without requiring semantic embedding models, making it suitable for large-scale evaluation pipelines.

F.3 Group Construction in GRPO-style Comparison

We adopt a group-based comparison strategy inspired by Group Relative Policy Optimization (GRPO), where multiple responses generated from the same prompt are evaluated jointly rather than independently.

Symbol	Parameter	Value	Module
K	Number of traces	5	I
T	Sampling temperature	0.7	I
α	TF-IRF importance floor	0.01	I
β	TF-IRF threshold	0.3	I
γ	Z-score anomaly cutoff	-1.0	II
λ	Edge weight balance	0.5	II
θ	Edge weight threshold	0.3	II

Table 7: Hyperparameter values. All symbols correspond to Algorithm 1. Thresholds β and θ are expressed as fractions of K.

For each input query, we sample a group of k reasoning traces:

$$\mathcal{G} = \{r_1, r_2, \ldots, r_k\}.$$

Instead of assigning absolute quality scores, responses are compared relative to other members in the same group. This relative comparison improves evaluation stability and reduces sensitivity to noise in individual traces.

This group comparison mechanism follows the core intuition of GRPO-style learning and evaluation: reasoning quality is more reliably estimated through intra-group comparison than through isolated scoring. It also enables scalable evaluation without requiring expensive reward models.

G Hyperparameter Settings

Table 7 lists all hyperparameters used in Algorithm 1 and their values for the experiments reported in Tables 2 and 4. The K-sensitivity analysis is in Section 5; all other values were fixed throughout.

Selection strategy. K=5 and T=0.7 were chosen to balance trace diversity against cost (Section 5). The two consensus thresholds (β, θ) are both set to 0.3 (i.e. an element must appear in $\geq$30% of traces), following the intuition that a step or edge supported by fewer than roughly one-third of independent rollouts is unlikely to reflect reliable reasoning. The z-score cutoff γ=-1.0 removes steps more than one standard deviation below the group mean in term-overlap; this is a moderate setting between the aggressive -0.5 and the conservative -1.5. λ=0.5 assigns equal weight to LLM-reported confidence and Jaccard similarity in the edge weight formula, reflecting that both signals are comparably reliable for determining edge quality. Synthesis uses temperature 0.0 (greedy decoding) because the diversity budget

Dataset	Original	Pass 1 (z-score) Del	Pass 1 (z-score) %	Pass 2 (RKG) Del	Pass 2 (RKG) %
FLD (nano)	4,465	327	7.3	1,338	32.3
FLD (o4-mini)	11,725	1,225	10.4	2,341	22.3
FOLIO (nano)	17,049	2,543	14.9	3,212	22.1
FOLIO (o4-mini)	16,835	2,478	14.7	3,843	26.8
GSM8K (nano)	2,781	1,375	49.4	253	8.8
GSM8K (o4-mini)	4,781	2,528	52.9	394	14.7
Olympiad (nano)	3,726	1,401	37.6	422	9.9
Olympiad (o4-mini)	5,892	2,793	47.4	661	18.0

Table 8: Step removal by filtering pass. *Original*: total reasoning steps across all traces (K=5). Pass 1 %: fraction of original steps removed by z-score. Pass 2 %: fraction of nodes and edges removed by RKG filtering.

Dataset	Filtering	Nodes Filtering	Edges Filtering
FLD (nano)	1,338	680 (47%)	758 (53%)
FLD (o4-mini)	2,341	683 (24%)	1,658 (76%)
FOLIO (nano)	3,212	1,209 (38%)	2,003 (62%)
FOLIO (o4-mini)	3,843	1,438 (38%)	2,405 (62%)
GSM8K (nano)	253	80 (32%)	173 (68%)
GSM8K (o4-mini)	394	111 (28%)	283 (72%)
Olympiad (nano)	422	87 (21%)	335 (79%)
Olympiad (o4-mini)	661	277 (42%)	384 (58%)

Table 9: RKG structural filter breakdown. *Nodes Filtering*: (zero in/out-degree) and *Edges Filtering*: weighted score lower than θ.

is spent in Module I; deterministic generation in Module III maximises faithfulness to the reference anchors.

H Filtering Statistics

Table 8 reports step removal rates for each filtering pass. Module II's z-score filter (Pass 1) removes 7–15% of steps on logical benchmarks and 38–53% on mathematical benchmarks, where vocabulary divergence is a stronger signal. The RKG structural filter (Passes 2+3) provides substantial additional filtering on logical benchmarks (22–32% extra removal, accounting for 56–80% of all deletions), but contributes less on mathematical benchmarks (9–22% extra).

Table 9 breaks down Pass 2 removals by anomaly type. Edge filtering (low-weight edges) dominates on most benchmarks (53–79%), while node filtering (isolated nodes with zero in/out-degree) accounts for 21–47% of removals.

I Case Study: Correct Label-Prediction, Flawed Reasoning Traces

We provide two examples drawn from actual LLM outputs in our pipeline: one from logical reasoning (FOLIO) and one from mathematical reasoning (OlympiadBench), demonstrating that the problem depicted in Figure 1 arises across both domains.

Example 1 — Step Internal Flaw (FOLIO, Logical Reasoning). *Problem:* Fact 1: TS Leda was a good passenger and cargo vessel. Fact 2: TS Leda was a Norwegian vessel that was built with stabilizers. Fact 3: Stabilizers are found only on ships with powerful steam turbine engines. Fact 4: To be a good passenger/cargo vessel, ships must be quiet and good at sea. Fact 5: *Some* ships that are quiet and good at sea have powerful steam turbine engines. Fact 6: Vessels are ships. **Hypothesis:** TS Leda was *not* a Norwegian vessel. **Ground truth: DISPROVED.**

LLM-generated trace (Trace A, 6 steps)

Step 1: From Fact 1, TS Leda was a good passenger and cargo vessel. ✓
Step 2: From Fact 4, TS Leda must be quiet and good at sea. ✓
Step 3: From Fact 5, *some* quiet/seaworthy ships have powerful steam turbine engines. Using Step 2, this implies TS Leda has powerful steam turbine engines. ✗ *quantifier fallacy* ($\exists \rightarrow \forall$)
Step 4: From Fact 3, stabilizers are found only on ships with powerful steam turbine engines. Since TS Leda has such engines (Step 3), it can have stabilizers. ✗ *propagates Step 3 error*
Step 5: From Fact 2, TS Leda was a Norwegian vessel with stabilizers. ✓
Step 6: Hypothesis says not Norwegian, but Step 5 (Fact 2) directly contradicts this. ✓
Label: DISPROVED ✓

The final label is correct, but Step 3 commits a **Step Internal Flaw**: Fact 5 states only that *some* quiet/seaworthy ships have powerful steam turbine engines ($\exists$), yet the trace infers that TS Leda specifically belongs to that set ($\forall$). This existential-to-universal fallacy propagates into Step 4. The correct label is reached only because Step 5 independently invokes Fact 2 — the flawed Steps 3–4 are logically superfluous. If used for distillation, the spurious $\exists \rightarrow \forall$ inference would be reinforced as a learned reasoning pattern.

Example 2 — Step-wise Flaw (OlympiadBench, Mathematical Reasoning). *Problem:* Define a *reverse prime* as a positive integer N whose digit-reversal is prime (e.g., 16 is a reverse prime since 61 is prime). Compute the largest two-digit integer

N such that N, $4N$, and $5N$ are all reverse primes.
Ground truth: $N = 79$.

Trace B (25 steps)

Step 1: Write $N = 10a + b$; define $R(M)$ as the digit-reversal of M. Need $R(N)$, $R(4N)$, $R(5N)$ all prime. ✓
Step 2: Since $N \leq 99$, note $5N \leq 495$ (three digits). Primes cannot end in 4, so $R(5N)$ must not end in 4. ✓
Step 3: Digit parity: for $R(N)$ to be prime and >5, its units digit must be odd, so the tens digit a of N is odd: $a \in \{1, 3, 5, 7, 9\}$. ✓
Step 4: Similarly restrict $4N$ and $5N$ by parity of their reversal's units digit. ✓
Step 5: Try $N = 99$: $R(99) = 99 = 9 \times 11$, not prime. ✗ *redundant*
Step 6: Try $N = 97$: $5 \times 97 = 485$, $R(485) = 584$, even — not prime. ✗ *redundant*
Steps 7–23: Try $N = 96, 95, 94, \ldots, 80$ one by one (each fails ≥ 1 condition). ✗ *redundant* $\times 17$
Step 24: Try $N = 79$: $R(79) = 97$ (prime), $4 \times 79 = 316$, $R(316) = 613$ (prime), $5 \times 79 = 395$, $R(395) = 593$ (prime). ✓
Step 25: Conclude $N = 79$ is the largest. ✓
Answer: $N = 79$ ✓

Trace B reaches the correct answer but commits a **Step-wise Flaw** (overthinking). Steps 1–4 derive digit-parity constraints that already narrow the candidate space; yet Steps 5–23 discard those results and enumerate every integer from 99 down to 80 one by one — 19 redundant steps that add no new reasoning. The efficient path needs only the bound $5N < 400 \Rightarrow N \leq 79$ (one deduction) plus three primality checks. In CRAFT, Steps 5–23 would be suppressed: their idiosyncratic per-value tokens (*try* $N = 96$, $N = 95$, ...) carry high within-trace TF but near-zero cross-trace IRF, yielding low TF-IRF scores and near-zero RKG edge frequency.

J Confidence Intervals

Table 10 reports 95% Wilson score confidence intervals for all accuracy values in Table 2. CRAFT achieves the widest margin on FLD under o4-mini (75.6 ± 3.8) and FOLIO under GPT-5.4-nano (89.6 ± 2.7), where CIs do not overlap with most baselines. On easier benchmarks (GSM8K, FOLIO under o4-mini) where baselines already achieve high accuracy, CRAFT's CIs partially overlap with the strongest baselines (e.g., Best-of-N on GSM8K), indicating that ceiling effects limit differentiation.

K Baseline & Computational Costs

We compare CRAFT against thirteen baselines that cover five broad strategy families: *voting/selection*, *iterative refinement*, *search-based*, *symbolic decomposition*, and *structural/causal trace analysis*. All multi-sample baselines use K=10 candidate traces. As shown in Table 11, CRAFT generates only K=5 traces but incurs additional calls for RKG construction and step-wise synthesis, totalling ≈15 per sample; setting baselines to K=10 therefore yields a comparable total API call budget. All baselines share the same backbone model (GPT-5.4-nano or o4-mini) at temperature T=0.7.

Self-Consistency (Wang et al., 2023) samples K reasoning traces and selects the final answer by majority vote, exploiting the intuition that correct reasoning paths are more likely to converge on the same answer. We use K=10 traces at T=0.7, following the original paper's recommended temperature range.

Universal Self-Consistency (USC) (Chen et al., 2023) extends Self-Consistency by replacing hard majority vote with an LLM-based selector: the model reads all K candidate answers and picks the most consistent one. This avoids format-sensitive answer extraction but adds one LLM call. We use K=10.

Self-Aggregation (Venkatraman et al., 2026) recursively aggregates K candidate traces into a single refined answer through iterative LLM summarisation rounds, rather than selecting one trace. We use K=10 candidate traces.

Best-of-N (Stiennon et al., 2022) generates K=10 candidate traces and selects the one with the highest self-evaluated score (the LLM rates its own traces). Unlike voting methods, it picks a single trace rather than aggregating answers.

Self-Refine (Madaan et al., 2023) iteratively improves a single trace through feedback–refinement loops: the model critiques its own output and revises it. We run 10 refinement iterations, providing ample budget for convergence.

Self-Eval Beam Search (Xie et al., 2023) generates reasoning traces step-by-step, using the

Setting	FLD	FOLIO	GSM8K	OlympiadBench
GPT-5.4-nano				
Best-of-N	56.0 ± 4.4	86.0 ± 3.0	94.8 ± 2.0	70.0 ± 4.0
Self-Consistency	61.8 ± 4.2	80.0 ± 3.5	93.0 ± 2.3	65.3 ± 4.2
Univ. Self-Consistency	61.4 ± 4.2	70.0 ± 4.0	92.0 ± 2.4	58.4 ± 3.2
Self-Refine	63.4 ± 4.2	86.0 ± 3.0	91.4 ± 2.5	63.4 ± 4.2
Self-Eval Beam Search	51.8 ± 4.4	78.0 ± 3.6	79.0 ± 3.6	25.4 ± 3.8
Faithful CoT	56.8 ± 4.3	68.6 ± 4.1	90.2 ± 2.6	51.0 ± 4.4
Tree-of-Thought	24.0 ± 3.1	74.0 ± 3.6	90.6 ± 2.0	46.6 ± 3.6
RAP	61.4 ± 3.8	78.6 ± 3.6	95.8 ± 2.2	55.6 ± 3.2
ConCISE	69.4 ± 3.8	83.2 ± 3.4	94.4 ± 2.2	63.2 ± 3.2
LCoT2Tree	69.4 ± 3.8	85.4 ± 3.6	94.8 ± 1.8	65.6 ± 4.0
DICE	70.0 ± 4.2	86.6 ± 4.3	95.0 ± 1.6	68.8 ± 4.0
PNS-Optimization	69.8 ± 4.0	83.6 ± 3.6	95.0 ± 2.4	67.4 ± 4.2
Self-Aggregation	68.4 ± 4.1	83.2 ± 3.2	93.0 ± 2.3	56.0 ± 4.4
CRAFT (Ours)	**71.6** ± 3.9	**89.6** ± 2.8	**96.0** ± 1.8	**73.8** ± 3.8
o4-mini				
Best-of-N	44.0 ± 4.2	82.8 ± 3.3	98.0 ± 1.3	64.0 ± 4.2
Self-Consistency	60.6 ± 4.2	82.0 ± 3.4	95.6 ± 1.8	68.0 ± 4.1
Univ. Self-Consistency	56.2 ± 4.4	82.0 ± 3.4	98.5 ± 1.1	57.6 ± 3.4
Self-Refine	44.8 ± 4.4	76.0 ± 3.8	93.2 ± 2.2	68.0 ± 4.1
Self-Eval Beam Search	62.0 ± 4.2	76.0 ± 3.8	37.8 ± 4.2	40.0 ± 4.4
Faithful CoT	48.0 ± 4.4	86.2 ± 5.2	93.2 ± 2.2	60.8 ± 4.2
Tree-of-Thought	52.4 ± 3.8	78.6 ± 3.8	91.8 ± 2.0	46.0 ± 4.3
RAP	70.4 ± 4.0	80.2 ± 3.6	95.0 ± 3.0	66.5 ± 4.3
ConCISE	66.4 ± 4.0	82.2 ± 3.6	93.4 ± 2.4	63.6 ± 3.8
LCoT2Tree	71.4 ± 3.8	85.4 ± 3.2	97.4 ± 2.4	65.6 ± 4.0
DICE	72.6 ± 4.2	85.2 ± 4.2	96.0 ± 1.8	68.8 ± 4.3
PNS-Optimization	71.6 ± 4.0	85.4 ± 4.2	94.6 ± 2.0	72.2 ± 4.2
Self-Aggregation	46.8 ± 4.4	80.0 ± 3.6	96.6 ± 1.6	62.4 ± 4.2
CRAFT (Ours)	**75.6** ± 3.8	**88.8** ± 2.8	**98.0** ± 1.4	**73.2** ± 4.0

Table 10: 95% Wilson confidence intervals for Table 2 accuracies. Format: Acc (%) ± half-width.

model's self-evaluation scores to prune and expand a beam of partial traces. We set beam width =2, following the original paper.

Faithful CoT (Lyu et al., 2023) decomposes reasoning into two stages: the LLM first translates the problem into a symbolic representation (e.g., Python or logic program), then executes it to derive the answer. This grounds reasoning in formal computation but assumes the problem is faithfully translatable.

RAP (Reasoning via Planning) (Hao et al., 2023) frames reasoning as planning with a world model: the LLM simulates future states and uses Monte Carlo Tree Search (MCTS) to explore the reasoning space, balancing exploration and exploitation via UCB scores.

Tree-of-Thought (ToT) (Yao et al., 2023) structures reasoning as a tree search where each node is a partial reasoning state. The model generates multiple next-step candidates (branching width =5), self-evaluates each, and selects the most promising branch up to depth =2. Parameters follow the original paper without per-task tuning.

ConCISE (Qiao et al., 2025) targets *overthinking* in reasoning traces by injecting confidence-guided cues that encourage the model to commit to high-confidence intermediate conclusions and suppress redundant reflection or backtracking, yielding shorter traces without sacrificing accuracy. We follow the authors' recommended prompting configuration and apply it per-sample at T=0.7.

LCoT2Tree (Jiang et al., 2025) converts a set of linear chain-of-thought traces into a merged tree in which semantically equivalent steps collapse into shared nodes, and uses graph-level structural features (depth, branching, consensus) to score and select the most promising reasoning path. It captures structural cues that linear majority voting cannot. We use K=10 candidate traces at T=0.7.

Module	Operation	Calls	Model
I	Generate K traces	K	backbone
	TF-IRF term extraction	0	—
II	Z-score anomaly filter	0	—
	Build per-trace RKGs	K	backbone
	Consensus RKG + structural filter	0	—
III	Topology-guided synthesis + verification	n^* + [0–2]	backbone
	Total per sample	$2K + n^* + [0\text{–}2]$	

Table 11: LLM API calls per sample. K=5; n^* is the number of non-fact nodes in the consensus RKG (typically 6–11). Total: ≈15 calls per sample. All modules use the same backbone model (GPT-5.4-nano or o4-mini).

DICE (Li et al., 2025) imposes an explicit structural scaffold on reasoning: the LLM decomposes each problem into sub-components, solves them independently, and recomposes the sub-solutions with intermediate verification, so that local errors can be caught and corrected before they propagate. Unlike voting-based methods, DICE operates on a single trace but enforces structured intermediate checks; we use the default decomposition depth from the original paper.

PNS-Optimization (Yu et al., 2025) scores each reasoning step by its causal contribution to the final answer using the *Probability of Necessity and Sufficiency* (PNS) estimator, and retains or re-weights steps according to their PNS values. This casts step-wise filtering as a causal-inference problem, improving the signal-to-noise ratio of candidate chains. We follow the paper's default PNS threshold and use K=10 candidate traces at T=0.7.

Cost comparison. TF-IRF extraction, z-score filtering, and graph pruning are pure computation and require zero LLM calls; the two LLM-heavy steps are trace generation (K calls) and per-trace RKG construction (K calls). With K=5 and a typical consensus RKG of ∼8 nodes, CRAFT totals ≈16 calls per sample—comparable to multi-sample baselines at K=10: Self-Consistency and Best-of-N make ≈10–20 calls, Self-Refine runs ∼16 feedback loops, and Self-Eval Beam Search uses $\sim K \times B$ calls for beam width B. CRAFT's overhead over simple voting methods comes from RKG extraction and stepwise synthesis (n^* calls), which directly account for the quality gains.

L Significance Testing Methodology

Figure 3 reports paired Wilcoxon signed-rank tests with 95% bootstrap confidence intervals. Because each panel aggregates multiple evaluation metrics into a single test, we describe the pooling procedure below.

Metric pooling. For a given model and subcategory (e.g., PRMBench *Soundness*), let N denote the number of evaluation items. Each item i yields a w/ Answer score $g_i^{(m)}$ and a w/o Answer score $b_i^{(m)}$ for each metric $m \in \mathcal{M}$. Rather than testing each metric independently, we concatenate the paired observations across all $|\mathcal{M}|$ metrics into a single pooled vector of size $N \times |\mathcal{M}|$:

$$\mathbf{d} = \big(g_1^{(1)} - b_1^{(1)},\ \ldots,\ g_N^{(1)} - b_N^{(1)},\ \ldots,\ g_1^{(|\mathcal{M}|)} - b_1^{(|\mathcal{M}|)},\ \ldots,\ g_N^{(|\mathcal{M}|)} - b_N^{(|\mathcal{M}|)}\big)$$

The Wilcoxon test and bootstrap CI are then computed on $\mathbf{d}$.

Metrics used. *PRMBench* (top row): $\mathcal{M}$ = {StepAcc, 1stErr, F1}, pooled per error dimension (Simplicity, Soundness, Sensitivity). *ROSCOE* (bottom row): $\mathcal{M}$ = {Faith., Info-Step, Info-Chain, Coher.}, pooled per dataset (CosmosQA, DROP, eSNLI, GSM8K).

Scale considerations. All pooled metrics are bounded in $[0, 1]$ and operate on comparable scales (accuracy or similarity-based), so raw pooling does not introduce scale dominance. The Wilcoxon signed-rank test is rank-based and therefore invariant to monotone transformations, further mitigating scale sensitivity.

Statistics. For each pooled vector $\mathbf{d}$: (1) mean difference and 95% CI via 5,000 bootstrap resamples; (2) two-sided Wilcoxon signed-rank test (p-value); (3) paired Cohen's $d = \bar{d}/s_d$. Significance markers: $*\,p < 0.05$, $**\,p < 0.01$, $***\,p < 0.001$.

M Experimental Prompt Templates

System Prompt (Prefix)

You are a meticulous logician. Produce exhaustive, atomic reasoning. Each step must cite the facts or earlier steps it depends on. Avoid circular references and avoid skipping steps.

M.1 RKG Edge Extraction (Module II)

BuildRKG Dependency Extraction Prompt

You are analyzing a step-by-step reasoning trace. Your task is to identify the DIRECT dependencies between steps.
For each reasoning step, identify which facts or EARLIER steps it DIRECTLY depends on to reach its conclusion.

- Only list DIRECT dependencies (not transitive ones).
- A step depends on another step if it uses that step's conclusion as a premise.
- Do NOT list a step as depending on itself.
- If a step uses no earlier steps or facts, set "uses" to [].

Output ONLY valid JSON with this exact structure:
`{"dependencies": [{"step_id": "Step1", "uses": []}, {"step_id": "Step2", "uses": ["Fact1", "Step1"]}, ...]}`
Available facts (given premises): {facts_block}
Reasoning steps: {steps_block}

M.2 Reasoning Traces Generation (w/o Answer)

Input Prompt - FLD & FOLIO Dataset

Generate extremely detailed reasoning for the following problem.
Problem Statement: {problem}
Instructions:

- Provide reasoning steps in the format "Step 1:", "Step 2:", etc.
- Each step must cite the exact facts or previous steps it uses.
- Use natural language sentences without JSON or markdown code fences.
- After the steps, include a short summary paragraph.
- Do not stop early; if the answer is cut off, immediately continue until the summary and final conclusion are delivered.
- Determine whether the hypothesis is `__PROVED__`, `__DISPROVED__`, or `__UNKNOWN__` based on your reasoning.
- End with a single line exactly formatted as: `Final Conclusion: __PROVED__` (or `__DISPROVED__`/`__UNKNOWN__`).

Input Prompt - LogiQA Dataset

Generate extremely detailed reasoning for the following problem.
Problem Statement: {problem}
Instructions:

- Provide reasoning steps in the format "Step 1:", "Step 2:", etc.
- Each step must cite the exact facts or previous steps it uses.
- Use natural language sentences without JSON or markdown code fences.
- After the steps, include a short summary paragraph.
- Do not stop early; if the answer is cut off, immediately continue until the summary and final conclusion are delivered.
- Select correct answers from options (A, B, C, or D) based on your reasoning.
- End with a single line exactly formatted as: `Answer: A` (or B / C / D).

Input Prompt - Multi-LogiEval Dataset

Generate extremely detailed reasoning for the following problem.
Problem Statement: {problem}
Instructions:

- Provide reasoning steps in the format "Step 1:", "Step 2:", etc.
- Each step must cite the exact facts or previous steps it uses.
- Use natural language sentences without JSON or markdown code fences.
- After the steps, include a short summary paragraph.
- Do not stop early; if the answer is cut off, immediately continue until the summary and final conclusion are delivered.
- Determine whether the answer is `yes` or `no` based on your reasoning.
- End with a single line exactly formatted as: `Answer: yes` (or `no`).

M.3 Reasoning Traces Generation (w/ Answer)

Input Prompt - FLD & FOLIO Dataset

Generate extremely detailed reasoning for the following problem.
Problem Statement: {problem}
Instructions:

- Provide 6–10 numbered steps in the format "Step 1:", "Step 2:", etc.
- Each step must cite the exact facts or previous steps it uses.
- Use natural language sentences without JSON or markdown code fences.
- After the steps, include a short summary paragraph.
- Do not stop early; if the answer is cut off, immediately continue until the summary and final conclusion are delivered.
- The ground truth conclusion is `{target_answer}`.
- Generate step-by-step reasoning that justifies this conclusion.
- End with: `Final Conclusion: {target_answer}`

Note: The correct conclusion for this problem is `{target_answer}`. Generate reasoning that validates this conclusion. Your final conclusion should match `{target_answer}`.

Input Prompt - LogiQA Dataset

Generate extremely detailed reasoning for the following problem.
Problem Statement: {problem}
Instructions:

- Provide reasoning steps in the format "Step 1:", "Step 2:", etc.
- Each step must cite the exact facts or previous steps it uses.
- Use natural language sentences without JSON or markdown code fences.
- After the steps, include a short summary paragraph.
- Do not stop early; if the answer is cut off, immediately continue until the summary and final conclusion are delivered.
- The ground truth answer is `{target_answer}`.
- Generate step-by-step reasoning that justifies this answer.
- End with: `Answer: {target_answer}`

Note: The correct answer for this problem is `{target_answer}`. Generate reasoning that validates this answer. Your final answer should be `{target_answer}`.

Input Prompt - Multi-LogiEval Dataset

Generate extremely detailed reasoning for the following problem.
Problem Statement: {problem}
Instructions:

- Provide reasoning steps in the format "Step 1:", "Step 2:", etc.
- Each step must cite the exact facts or previous steps it uses.
- Use natural language sentences without JSON or markdown code fences.
- After the steps, include a short summary paragraph.
- Do not stop early; if the answer is cut off, immediately continue until the summary and final conclusion are delivered.
- The ground truth answer is `{target_answer}`.
- Generate step-by-step reasoning that justifies this answer.
- End with: `Answer: {target_answer}`

Note: The correct answer for this problem is `{target_answer}`. Generate reasoning that validates this answer. Your final answer should be `{target_answer}`.

M.4 Reasoning Traces Evaluation Prompt Template

Holistic Classification Prompt

Please evaluate the quality distribution of the following reasoning process with BALANCED and NUANCED judgment.
Problem Context: {problem_input}
Reasoning Process: {reasoning_process}
Expected Answer/Ground Truth: {ground_truth}
Evaluation Requirements:
IMPORTANT EVALUATION PRINCIPLES:

1. Be STRICT with "logical_error" – only assign high percentages (>50%) for SERIOUS, CLEAR logical flaws
2. Be GENEROUS with other categories – minor issues should be distributed across multiple categories
3. AVOID extreme values – aim for 2–3 non-zero categories in most cases
4. Most real reasoning has MIXED quality – reflect this in your distribution

Category Definitions (with strict/generous guidelines):

- **perfect**: All steps correct, optimal path
- **logical_error**: ONLY for CLEAR, SERIOUS invalid reasoning. Minor imperfections are NOT logical errors!
- **step_missing_rate**: Missing intermediate steps, gaps in reasoning.
- **early_termination_rate**: Stopped before reaching conclusion.
- **redundant**: Unnecessary steps, non-optimal path.
- **hallucination**: References facts that don't exist in the problem context

Important:

- Content distribution (first 6) must sum to exactly 100%
- Structure rates (last 1) are independent percentages, each can be 0–100%
- Consider the proportion of reasoning affected by each issue

Output Format (JSON):

```
{"distribution": {
   "perfect": <0-100>,
   "logical_error": <0-100>,
   "step_missing_rate": <0-100>,
   "early_termination_rate": <0-100>,
   "redundant": <0-100>,
   "hallucination": <0-100>},
 "structure_problems": {
   "wrong_order": <0-100>},
 "details": {
   "has_logical_error": true/false,
   "has_wrong_order": true/false,
   "has_hallucination": true/false,
   "has_redundancy": true/false,
   "has_missing_steps": true/false,
   "reached_conclusion": true/false}}
```

You are an expert in reasoning process quality evaluation. Your task is to analyze given reasoning processes and identify their quality types.
Category 1: Content-level Quality Types:

1. **Perfect**: All reasoning steps are logically correct, path is optimal
2. **Logical Error**: Contains logical reasoning errors
3. **Step Missing Rate**: Percentage of step transitions with missing steps (gaps, skipping)
4. **Early Termination Rate**: Percentage indicating premature termination
5. **Redundant**: Reasoning is correct but contains unnecessary intermediate steps, not the shortest path
6. **Hallucination**: References non-existent facts or fabricates information

Category 2: Structure-level Problems (independent rates, 0–100% each):

- **Wrong Order**: Steps are correct individually but in wrong order, wrong premise selection, or wrong conditional branch usage

Category 3: Informal Word Detection (keyword list matching 0–100%):

- **Informal Word Rate**: Percentage of steps containing emotional words (maybe, perhaps, etc.)

Please strictly follow the evaluation criteria and provide clear classification results.

M.5 Step-wise Evaluation Prompts

Logical Error Evaluation

Problem Context: {problem_input}
Reasoning Process: {reasoning_process}
Expected Answer: {ground_truth}
Evaluation Dimension: Logical Reasoning Error
Please check each reasoning step carefully with STRICT criteria:

- **True** only if there are CLEAR and SERIOUS logical errors where premises COMPLETELY FAIL to support the conclusion
- Minor imperfections, slightly informal reasoning, or small gaps are NOT logical errors
- A reasoning step with 80%+ correctness should be considered logically sound
- Focus on MAJOR errors: clear contradictions, completely wrong causal relationships, or blatantly false inferences

Examples of TRUE (serious logical errors):

- Premises say “A>B and B>C” but conclusion is “C>A” (completely wrong)
- Using “all X are Y” to conclude “all Y are X” (invalid reversal)
- Clear mathematical calculation errors that change the result significantly

Examples of FALSE (not serious enough for logical error):

- Slightly incomplete explanation but generally sound reasoning
- Missing some minor details but core logic is correct
- Informal language but logically valid inference

Be strict: Only answer true if there are CLEAR, UNDENIABLE logical errors. If the reasoning is mostly sound (>80% correct), answer false.
Answer only: true or false

Hallucination Evaluation

Problem Context: {problem_input}
Reasoning Process: {reasoning_process}
Expected Answer: {ground_truth}
Evaluation Dimension: Hallucination Content
Please check if the reasoning process references information that does not exist in the problem context, or fabricates new "facts" out of thin air. Note: Reasonable inference conclusions are NOT hallucinations, but referencing non-existent original facts IS hallucination.
Examples:

- Hallucination: Problem says "x>5", but reasoning says "Given x=10..." (fabricates specific value)
- Not hallucination: Infers "x is positive" from "x>5" (reasonable inference)

If hallucination exists, answer true; otherwise answer false.
Answer only: true or false

Informal Words Evaluation

We measure the seriousness of generated reasoning traces via keyword matching.

```
EMOTIONAL_WORDS = {
  "maybe", "perhaps", "possibly",
  "probably", "might", "may",
  "could", "would", "should",
  "seems", "appears", "looks like",
  "sort of", "kind of", "a bit",
  "quite", "rather", "very",
  "really", "actually", "basically",
  "essentially", "obviously", "clearly",
  "surely", "certainly", "definitely",
  "absolutely", "literally", "honestly",
  "frankly", "admittedly", "supposedly",
  "allegedly", "hopefully",
  "unfortunately", "luckily", "sadly",
  "surprisingly", "interestingly",
  "amazingly", "incredibly",
  "extremely", "pretty", "somewhat",
  "slightly"}
```

Step Missing Evaluation

Problem Context: {problem_input}
Reasoning Process: {reasoning_process}
Expected Answer: {ground_truth}
Evaluation Dimension: Missing / Skipped Steps
Please check if ANY transition between statements would benefit from an extra intermediate step.

- If the reasoning jumps from A to C and you think "hmm, I wish there was a short B in between", count it as missing
- Even if the writer hints at why it works, but you still feel an explicit bridge would help, count it as missing
- Small logical leaps, implicit assumptions, or "hand-wavy" transitions all count as missing steps
- Only answer false when every transition feels naturally justified with no gaps

If you feel additional steps would make it clearer, answer true; Only answer false when the chain already feels fully explicit.
Answer only: true or false

Redundancy Evaluation

Problem Context: {problem_input}
Reasoning Process: {reasoning_process}
Expected Answer: {ground_truth}
Evaluation Dimension: Redundant / Wordy / Overly Verbose Steps
Please check if ANY part of the reasoning feels redundant, wordy, or unnecessary – even if it still reaches a correct conclusion. Use a LENIENT standard:

- Repeating the same sentence/idea with slightly different wording counts as redundant
- Taking a detour when a shorter path is available counts as redundant
- Adding filler sentences like "Clearly, obviously, we restate..." counts as redundant
- If you personally feel "this sentence is extra / could be removed without loss", then it is redundant

If you sense redundancy anywhere, answer true; Only answer false when every sentence feels essential and concise.
Answer only: true or false

Wrong Order Evaluation

Problem Context: {problem_input}
Reasoning Process: {reasoning_process}
Expected Answer: {ground_truth}
Evaluation Dimension: Wrong Order / Premise Selection / Conditional Errors
Please check if the reasoning process has ordering or selection issues (including subtle ones):

- **Strict errors**: Steps use facts/conclusions before they are defined
- **Premise selection issues**: Wrong premise selection (choosing incorrect or suboptimal premises from available options)
  - Using only one fact when multiple relevant facts are available
  - Ignoring relevant intermediate conclusions that should be used
  - Choosing less relevant premises over more relevant ones
- **Ordering issues**: Steps are logically correct individually but in suboptimal sequence
  - Steps that could be executed earlier are placed later
  - Steps that should follow a logical sequence are out of order
- **Conditional branch errors**: Wrong conditional branch usage (using incorrect if-then branches)

Examples:

- Strict: Using int2 before int1 is defined (wrong order)
- Premise selection: From fact1, fact2, fact3, choosing only fact1 when fact2 and fact3 are also relevant
- Ordering: Step 5 could have been done at step 2, but was delayed unnecessarily
- Conditional: Using "if P then Q" when "if not P then R" is the correct branch

Relaxed criteria: If there are ANY signs of ordering, selection, or conditional issues (even subtle ones), answer true. If steps are in optimal order and use the most appropriate premises, answer false.
Answer only: true or false

Early Termination Evaluation

Problem Context: {problem_input}
Reasoning Process: {reasoning_process}
Expected Answer: {ground_truth}
Evaluation Dimension: Early Termination
Please check whether the reasoning genuinely ends where it should or if it stops earlier than necessary.

- Answer true only when you feel additional reasoning is required to finish the task (e.g., explicitly quits, keeps gathering information, or clearly states it cannot proceed yet).

- Answer false when the reasoning naturally concludes or the task itself is satisfied, even if the answer is short or lacks an explicit "Conclusion" sentence.
- Do NOT use the absence of a conclusion keyword as evidence; focus on whether the reasoning itself signals incompleteness.

Answer only: true or false

M.6 Step-level Evaluation Prompt

Premise-Conclusion Entailment Evaluation

Premises: {premise1} {premise2} ...
Conclusion: {conclusion}
Do the premises entail the conclusion? Answer true or false only.

Atomic Inference Evaluation

Premises: {premise1} {premise2} ...
Conclusion: {conclusion}
Is this inference atomic? Answer true or false only.

Completion Evaluation

Problem Context: {problem_input}
Reasoning Process: {reasoning_process}
Task: Determine whether the reasoning naturally reaches its intended stopping point or if it appears truncated / incomplete.
Guidelines:

- Answer true ONLY if the reasoning clearly indicates it could/should continue but stops early (e.g., explicitly says it cannot continue, still collecting evidence, or obviously leaves the core question unresolved).
- Answer false if the reasoning feels naturally complete for the task, even if no explicit "Conclusion" sentence is present.
- Do NOT use the absence of conclusion keywords as evidence; base the decision on whether the reasoning itself signals incompleteness.
- Consider whether the final step actually performs the last required action. If it already evaluates the hypothesis or exhausts the given facts, treat it as complete.

Answer only: true or false

M.7 Common Reasoning Trace Terms, High *TF*, Low *IRF*

Lexicon Definitions

LOGICAL_KEYWORDS:

- **Step-related**: step, steps, step1, step2, step3, step4, step5, first, second, third, next, then, finally.
- **Conditional**: if, else, elif, when, whenever, unless, provided.
- **Causality**: because, since, as, due, therefore, thus, hence, so, consequently, accordingly.
- **Connectors**: and, or, but, however, moreover, furthermore, additionally, also, besides.
- **Inference**: implies, imply, conclude, conclusion, infer, inference, deduce, deduction.
- **Proof-related**: prove, proof, proven, disprove, disproven, contradiction, contradictory, assume, assumption.
- **Fact-referencing**: fact, facts, given, premise, premises, hypothesis, statement, claim.
- **Logical Operations**: all, any, some, none, every, each, not, no, never, always.
- **Comparison**: equal, equals, equivalent, same, different, greater, less, than, from.

COMMON_LOGICAL_WORDS (High-frequency):
because, also, so, and, or, but, if, then, therefore, thus, since, as, when, not, no, all, any, some, each, every, fact, facts, step, steps, prove, proof, conclude, conclusion, approve, disprove, given, statement, claim, implies, however.
STOPWORDS:
the, a, an, is, are, was, were, be, been, being, have, has, had, do, does, did, will, would, could, should, may, might, must, can, this, that, these, those, it, its, they, them, their, there, here, where, what, which, who, whom, whose, how, why, when, to, of, in, on, at, by, for, with, from, as, into, onto, up, down, out, off, over, under, above, below, between, among, through, during, before, after, while, about, against, within, without, throughout, across, around, near, far, inside, outside, beside, besides, except, including, excluding, concerning, regarding, according, i, you, he, she, we, us, our, your, my, me, him, her, his, hers, mine, yours, ours, theirs.

N Baseline Prompt Templates

N.1 Direct Setting

System Prompt (Prefix)

You are a logical reasoning expert. Analyze the given problem and provide your conclusion directly without showing intermediate reasoning steps.

Input Prompt - FLD & FOLIO Dataset

{problem}
Based on the facts and hypothesis above, determine whether the hypothesis is:

- `__PROVED__`: The hypothesis can be logically proven from the facts.
- `__DISPROVED__`: The hypothesis can be logically disproven from the facts.
- `__UNKNOWN__`: The hypothesis cannot be determined from the given facts.

Provide your answer directly in the format: `Final Conclusion: __PROVED__` (or `__DISPROVED__` / `__UNKNOWN__`).

Input Prompt - LogiQA Dataset

{problem}
Based on the context and question above, select the correct answer from the options (A, B, C, or D).
Provide your answer directly in the format: `Answer: A` (or B / C / D).

N.2 Chain of Thought (CoT) Setting

System Prompt (Prefix)

You are a meticulous logician. Analyze the problem step by step using chain of thought reasoning. Each step should build on the previous ones to reach a logical conclusion.

Input Prompt - FLD&FOLIO

{problem}
Think step by step to solve this problem.
Instructions:

1. Think through this problem step by step.
2. Use numbered steps in the format "Step 1:", "Step 2:", etc.
3. Each step should cite the facts or previous steps it uses.
4. Build your reasoning logically from one step to the next.
5. After your reasoning steps, provide your final conclusion.

Based on your step-by-step reasoning, determine whether the hypothesis is:

- `__PROVED__`: The hypothesis can be logically proven from the facts.

- `__DISPROVED__`: The hypothesis can be logically disproven from the facts.
- `__UNKNOWN__`: The hypothesis cannot be determined from the given facts.

End with: `Final Conclusion: __PROVED__` (or `__DISPROVED__` / `__UNKNOWN__`).

Input Prompt - LogiQA Dataset

{problem}
Think step by step to solve this logical reasoning problem.
Instructions:

1. Think through this problem step by step.
2. Use numbered steps in the format "Step 1:", "Step 2:", etc.
3. Each step should cite the context, question, or previous steps it uses.
4. Build your reasoning logically from one step to the next.
5. Evaluate each option carefully.
6. After your reasoning steps, provide your final answer.

Based on your step-by-step reasoning, select the correct answer from the options (A, B, C, or D).
End with: `Answer: A` (or B / C / D).

N.3 Tree-of-Thought (ToT) Setting

System Prompt (Prefix)

You are a logical reasoning expert using tree-of-thought methodology. Explore multiple reasoning paths, evaluate them, and select the best one.

Input Prompt - FLD & FOLIO Dataset

{problem}
Use tree-of-thought reasoning to solve this problem step by step:
Step 1: Brainstorm 2–3 different approaches to solve this problem. Briefly describe each approach (1–2 sentences each).
Step 2: Evaluate each approach. Consider: feasibility, logical soundness, and completeness. Identify the most promising approach.
Step 3: Expand the best approach into detailed reasoning. Continue with Step 4:, Step 5:, etc., building a logical chain. Each step should cite facts or previous steps. Continue reasoning until you reach a conclusion.
Based on your analysis, determine whether the hypothesis is:

- `__PROVED__`: The hypothesis can be logically proven from the facts.
- `__DISPROVED__`: The hypothesis can be logically disproven from the facts.
- `__UNKNOWN__`: The hypothesis cannot be determined from the given facts.

End with: `Final Conclusion: __PROVED__` (or `__DISPROVED__` / `__UNKNOWN__`).

Input Prompt - LogiQA Dataset

{problem}
Use tree-of-thought reasoning to solve this problem step by step:
Step 1: Brainstorm 2–3 different approaches to analyze this problem. Briefly describe each approach (1–2 sentences each).
Step 2: Evaluate each approach. Consider: feasibility, logical soundness, and completeness. Identify the most promising approach.
Step 3: Expand the best approach into detailed reasoning. Continue with Step 4:, Step 5:, etc., building a logical chain. Each step should cite the context, question, options, or previous steps. Continue reasoning until you reach a conclusion.
Based on your analysis, select the correct answer from the options (A, B, C, or D).
End with: `Answer: A` (or B / C / D).

N.4 Chain of Draft (CoD) Setting

System Prompt (Prefix)

You are a logical reasoning expert using Chain of Draft methodology. First quickly generate a draft answer, then verify and refine it.

Input Prompt - FLD&FOLIO Dataset

{problem}
Use Chain of Draft reasoning to solve this problem:
Step 1: Draft Conclusion Quickly generate a draft conclusion (`__PROVED__`, `__DISPROVED__`, or `__UNKNOWN__`) with minimal reasoning.
Format: `Draft Conclusion: [__PROVED__/__DISPROVED__/ __UNKNOWN__]`
Step 2: Verify and Refine Verify the draft conclusion by checking against the facts and hypothesis. If the draft is correct, confirm it. If not, revise it. Provide brief reasoning for your verification. Continue with Step 3:, Step 4:, etc. if needed for detailed verification.
Based on your verification, provide the final conclusion.
End with: `Final Conclusion: __PROVED__` (or `__DISPROVED__` / `__UNKNOWN__`).

Input Prompt - LogiQA Dataset

{problem}
Use Chain of Draft reasoning to solve this problem:
Step 1: Draft Answer Quickly generate a draft answer (A, B, C, or D) with minimal reasoning. Format: `Draft Answer: [A/B/C/D]`
Step 2: Verify and Refine Verify the draft answer by checking against the context and question. If the draft is correct, confirm it. If not, revise it. Provide brief reasoning for your verification. Continue with Step 3:, Step 4:, etc. if needed for detailed verification.
Based on your verification, provide the final answer.
End with: `Answer: A` (or B / C / D).

N.5 In-Context Learning (ICL) Setting

System Prompt (Prefix)

You are a logical reasoning expert. Learn from the examples and apply similar reasoning patterns to the new problem.

Input Prompt - FLD&FOLIO Dataset

Here are examples of different types of logical reasoning problems:
Example 1 ({label1}): Problem: {example1_problem}
Solution: {example1_proof}
—
Example 2 ({label2}): Problem: {example2_problem}
Solution: {example2_proof}
Example 3 ({label3}): Problem: {example3_problem}
Solution: {example3_proof}
Now solve this new problem using similar reasoning:
{problem}
Instructions:

1. Learn from the reasoning patterns shown in the examples.
2. Use numbered steps in the format "Step 1:", "Step 2:", etc.
3. Each step should cite the facts or previous steps it uses.
4. Apply the appropriate logical structure based on what the problem requires.
5. Note that not all problems can be proved or disproved – some remain unknown.

Based on your reasoning, determine whether the hypothesis is:

- `__PROVED__`: The hypothesis can be logically proven from the facts.

- `__DISPROVED__`: The hypothesis can be logically disproven from the facts.
- `__UNKNOWN__`: The hypothesis cannot be determined from the given facts.

After your reasoning steps, provide your answer in the format: `Final Conclusion: __PROVED__` (or `__DISPROVED__` / `__UNKNOWN__`).

Input Prompt - LogiQA Dataset

Here are examples of logical reasoning problems:
Example 1 (Answer: {label1}): Problem: {example1_problem}
Solution: {example1_proof}
—
Example 2 (Answer: {label2}): Problem: {example2_problem}
Solution: {example2_proof}
—
Example 3 (Answer: {label3}): Problem: {example3_problem}
Solution: {example3_proof}
—
Now solve this new problem using similar reasoning:
{problem}
Instructions:

1. Learn from the reasoning patterns shown in the examples.
2. Use numbered steps in the format "Step 1:", "Step 2:", etc.
3. Each step should cite the context, question, options, or previous steps it uses.
4. Apply the appropriate logical structure based on what the problem requires.
5. Evaluate each option carefully.

Based on your reasoning, select the correct answer from options (A, B, C, or D).
After your steps, provide your answer in the format: `Answer: A` (or B / C / D).

N.6 PRMBench Step Verifier

System Prompt — w/ Answer

You are a mathematical reasoning verifier. You will be given a math problem, its correct final answer, and numbered solution steps. For each step, output two scores:

- **validity**: $+1.0$ if the step is logically correct, -1.0 if it contains an error, or a value in between for partial errors.
- **redundancy**: $+1.0$ if the step is redundant (adds no new information), -1.0 if it is necessary and informative.

Respond with ONLY a JSON object: `{"validity": [s1, s2, . . . ], "redundancy": [s1, s2, . . . ]}`

Example:
Question: What is 15% of 80?
Solutions:
Step 1: To find 15% of 80, we convert the percentage to a decimal: $15\% = 0.15$.
Step 2: Now multiply: $0.15 \times 80 = 120$.
Step 3: Therefore the answer is 120.
Output: `{"validity": [1.0, -1.0, -1.0], "redundancy": [-1.0, -1.0, -1.0]}`
Explanation: Step 1 is valid. Step 2 has an arithmetic error ($0.15 \times 80 = 12$, not 120), so validity $= -1$. Step 3 propagates the wrong answer, so validity $= -1$.

User Prompt — w/ Answer

Question: {question}
Correct final answer: {correct_answer}
Solutions:
Step 1: {step_1}
Step 2: {step_2}

⋮
Output a JSON object with validity and redundancy arrays, each of length {n_steps}.

System Prompt — w/o Answer

You are a mathematical reasoning verifier. You will be given a math problem and numbered solution steps. For each step, output two scores:

- **validity**: $+1.0$ if the step is logically correct, -1.0 if it contains an error, or a value in between for partial errors.
- **redundancy**: $+1.0$ if the step is redundant (adds no new information), -1.0 if it is necessary and informative.

Respond with ONLY a JSON object: `{"validity": [s1, s2, . . . ], "redundancy": [s1, s2, . . . ]}`

Example:
Question: What is 15% of 80?
Solutions:
Step 1: To find 15% of 80, we convert the percentage to a decimal: $15\% = 0.15$.
Step 2: Now multiply: $0.15 \times 80 = 120$.
Step 3: Therefore the answer is 120.
Output: `{"validity": [1.0, -1.0, -1.0], "redundancy": [-1.0, -1.0, -1.0]}`
Explanation: Step 1 is valid. Step 2 has an arithmetic error ($0.15 \times 80 = 12$, not 120), so validity $= -1$. Step 3 propagates the wrong answer, so validity $= -1$.

User Prompt — w/o Answer

Question: {question}
Solutions:
Step 1: {step_1}
Step 2: {step_2}
⋮
Output a JSON object with validity and redundancy arrays, each of length {n_steps}.