

Understanding Failures in LLM Reasoning by Learning Structured Representations of Chain-of-Thought

Tommaso Felice Banfi* Mubashara Akhtar* Jingwei Ni Mrinmaya Sachan
ETH Zurich

{tbanfi, makhta, jingni, msachan}@ethz.ch

Abstract

Chain-of-Thought (CoT) is central to improving reasoning in large language models, yet existing evaluation methods focus on step-level signals and ignore the underlying structure of reasoning traces. This raises a key question: what aspects of reasoning quality (i.e., correctness, efficiency, and failure patterns) are encoded in the structure of a CoT trace? We introduce **CoT2Graph**, a framework that represents CoT traces as directed dependency graphs, making relationships between reasoning steps explicit and capturing patterns such as backtracking, branching, and subtask merging. We model these graphs with graph neural networks and use attribution-based explanations to identify structural patterns associated with reasoning quality. CoT2Graph reveals distinct patterns of successful and failed reasoning: correct CoTs place greater importance on early planning, whereas incorrect CoTs are associated with overloaded synthesis steps and dependency violations. We further show that graph-based signals better predict whether a CoT leads to a correct answer, outperforming previous tree-based structural methods by up to 2.5% and simple length-based baselines by 7.8%, while also improving Best-of- N selection. Overall, our results demonstrate that reasoning structure encodes rich signals about reasoning quality, motivating a shift from sequence-level evaluation to structural analysis.

 BanfiTommasoFelice/LLM-StructReasoning-EMNLP26

1 Introduction

Models trained with reinforcement learning on reasoning tasks produce long Chain-of-Thought (LCoT) traces that involve backtracking over earlier conclusions, branching across concurrent subproblems, and combining partial results from multiple reasoning threads (Wu et al., 2026). However, existing evaluation approaches treat these

CoT2Graph: CoT $\rightarrow$ Reasoning Structure

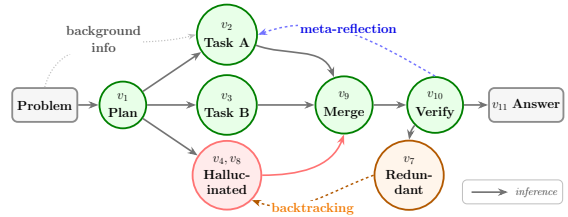


Figure 1: A CoT chain represented as an interpretable directed graph capturing the reasoning process.

traces either as black boxes, measuring only the final-answer correctness, or as linear sequences, scoring each step individually. This raises a central question: what aspects of *reasoning quality* such as correctness, efficiency, and failure patterns are encoded in the structure of a CoT trace? Step-level evaluation frameworks such as ROSCOE (Golovneva et al., 2023) and ReCEval (Prasad et al., 2023) detect local errors, but evaluate reasoning steps largely in isolation and therefore do not capture global reasoning structure. Existing structured representations based on sequences or trees (Dalvi et al., 2021; Jiang et al., 2025b) provide a more global view, but cannot represent multi-source dependencies, the merging of reasoning threads, or the revisitation and reuse of intermediate results. It remains unclear whether and how CoT structure encodes reasoning quality.

To address this question, we introduce **CoT2Graph**, a framework that represents CoT traces as directed dependency graphs (Figure 1) and models them using graph neural networks (GNNs). Nodes are reasoning steps, and edges encode relations such as *inference*, *repetition*, *observation*, and *meta-reflection*. The GNN predicts final-answer correctness from these typed dependencies, while GNNExplainer identifies the reasoning steps most decisive for each prediction. We provide new insights into reasoning behavior, identifying structural patterns of successful and failed reasoning

*Equal contribution

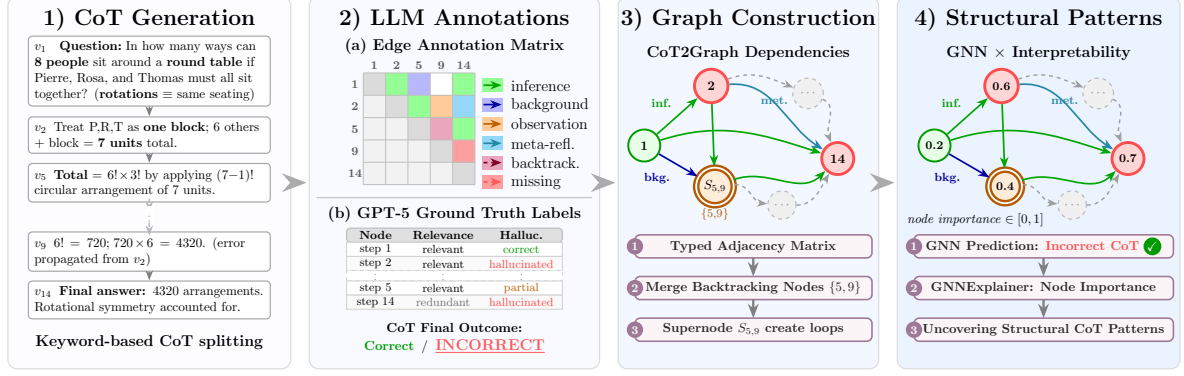


Figure 2: CoT2Graph pipeline for graph construction and reasoning structural analysis (GPQA ID 32; Figure 13).

traces. Correct CoTs place greater importance on early planning and task decomposition while incorrect CoTs, in contrast, are associated with overloaded synthesis steps that combine multiple reasoning threads and with violations of inferred dependencies, such as proceeding before prerequisite steps are generated. We also find that longer reasoning is not always more efficient: while longer traces contain more verification and exploration, they are often more inefficient. These findings suggest that reasoning quality depends not only on the correctness of steps, but also on how intermediate reasoning is organized and composed in the CoT.

Contributions:

- We introduce CoT2Graph, which maps CoT traces to typed dependency graphs (§3).
- Graph structure predicts correctness better than tree baselines, with gains up to 2.5% (§5.2).
- Attributions reveal failure patterns: correct traces emphasize early planning, while incorrect traces rely on overloaded synthesis nodes (§5.4).
- We decompose redundancy into verification-exploration patterns and path inefficiency, finding 49.0% average path-efficiency loss (§5.5).

2 Related Work

Structured Representations of Reasoning. Recent reinforcement learning and test-time scaling approaches have enabled LLMs to generate longer CoT traces, leading to substantial improvements in reasoning performance (Guo et al., 2025; Fu et al., 2025; Chen et al., 2025; Ni et al., 2026). However, most prior work focuses on improving performance rather than understanding the underlying reasoning structure. Reasoning representations have been studied with entailment trees (Dalvi et al., 2021), which organize multi-hop inference into hierarchi-

cal steps. Tree of Thoughts (Yao et al., 2023; Banfi and Gamage, 2026) and Graph of Thoughts (Besta et al., 2024) impose structured reasoning at inference time via branching search. Jiang et al. (2025b) take the latter approach with LCoT2Tree, converting generated CoT traces to hierarchical trees via LLM-annotated sketches. We extend this by using directed graphs that encode cycles and multi-source synthesis (inexpressible in a tree), while requiring a 7B model rather than a frontier LLM.

Step-Level Evaluation and Error Taxonomy.

Step-level semantic frameworks evaluate CoT quality beyond final accuracy: ROSCOE (Golovneva et al., 2023) covers alignment, coherence, and grammaticality; ReCEval (Prasad et al., 2023) assesses correctness and informativeness via NLI; and a recent survey (Lee and Hockenmaier, 2025) proposes a taxonomy of factuality, validity, coherence, and utility. Error analysis has been applied to mathematical (Li et al., 2024), legal (Mishra et al., 2025), and multimodal CoT (Jiang et al., 2025a), and verification methods correct intermediate errors via chain reversal (Xue et al., 2023), external knowledge editing (Zhao et al., 2023), and step-level verification (Xu et al., 2025; Jacovi et al., 2024). On efficiency, longer traces do not monotonically improve accuracy and can introduce redundancy (Wu et al., 2026; Xue et al., 2024). Our work adds two structural error categories to the semantic taxonomy, shows that step redundancy is predictable from CoT structure, and provides an efficiency decomposition that separates productive exploration-verification patterns from inefficiency.

3 CoT2Graph Framework

CoT2Graph converts Chain-of-Thought (CoT) traces into structured dependency graphs and mod-

els them using graph neural networks (GNNs) to analyze reasoning quality. The pipeline (Figure 2) consists of four stages: (i) generating CoT traces and segmenting them into atomic reasoning steps, (ii) annotating dependencies between nodes, and (iii) constructing a graph representation for (iv) interpretability analysis. This representation captures non-linear reasoning patterns such as branching, merging, and revisitation, while the GNN models support the analysis of reasoning quality, including correctness, efficiency, and failure modes.

3.1 Generation of CoT Traces

We generate long CoTs using three open-weight reasoning models: DeepSeek-R1-Distill-Qwen-7B (DS-Qw7B) (Guo et al., 2025), DeepSeek-R1-Distill-Qwen-14B (DS-Qw14B), and MiMo-7B-RL (Xiaomi LLM-Core Team, 2025). Traces are generated on the benchmarks GPQA and MMLU-Pro. To ensure comparability across models, we use a consistent setup with low temperature ($T = 0.1$) and a maximum generation of 16K tokens. This configuration produces stable yet sufficiently diverse CoT traces for downstream analysis.¹

3.2 Graph Representation

We represent each CoT trace as a directed graph $G = (V, E, \phi, \psi)$, where nodes $V = \{v_1, \dots, v_n\}$ represent atomic reasoning steps and edges $E \subseteq V \times V$ capture dependencies between them. Each edge is assigned a reasoning relation (i.e., *inference*, *background knowledge*, *observation*, *meta-reflection*, or *repetition*) and a strength score indicating the importance of the dependency.

For each node, the feature map stores its generation position and local topology, following the node features of LCoT2Tree (Jiang et al., 2025b).

Let $\mathcal{T}$ be the edge-type vocabulary. Each edge (v_i, v_j) is encoded by ψ as its type $\tau_{ij} \in \mathcal{T}$ in one-hot form, concatenated with its normalized strength s_{ij} . Together, ϕ and ψ transform a linear CoT into a typed, weighted dependency graph that exposes multi-step dependencies and non-linear reasoning patterns. To isolate structural signals, the predictor excludes CoT textual information and relies only on the CoT structure; step semantics are included as an ablation in Appx. D. This representation preserves the order of reasoning steps, allowing us to study whether the organization of reasoning provides predictive signal about reasoning quality.

¹Further details are provided in §5.1.

3.3 Node Segmentation

Each CoT trace is segmented into atomic reasoning steps $V = \{v_1, \dots, v_n\}$ using a rule-based parser that identifies structural transition markers (e.g., *Therefore*, *However*, and *Finally*) via regex matching. Fragments shorter than 100 characters are merged to produce meaningful reasoning units. The resulting sequence preserves the original generation order while producing coherent reasoning units for downstream analysis. Appx. B compares parser boundaries with GPT-5 Mini on GPQA and reports sensitivity across domains; DS-Qw7B reaches 91.5% agreement, and 86.7% of traces are within a 10% boundary-error tolerance.

3.4 Edge Annotation

We infer dependencies between reasoning steps using an instruction-tuned LLM (Llama-3.1-8B-Instruct). For each ordered pair of steps (v_i, v_j) with $i < j$, the model predicts (i) a dependency type and (ii) a strength/importance score $\in [0, 10]$, producing a typed directed edge.

The dependency type is selected from a fixed set of reasoning transitions: *inference* (v_j follows directly from v_i), *background* (v_i supplies knowledge or assumptions used in v_j), *observation* (v_j observes or comments on v_i), *meta-reflection* (v_j plans or checks the reasoning rather than advancing it), *repetition* (v_i and v_j express the same content), and *none* (no meaningful dependency). Appx. C gives the full definition and strength scale for each edge type, and relates the taxonomy to entailment, retrieval, and search-based reasoning literature. Restricting to pairs with $i < j$ ensures that edges follow the original generation order, i.e., dependencies are only assigned from earlier to later steps. This encodes a causal prior consistent with how CoT traces are produced, where each step conditions on previously generated content. The resulting edges capture how reasoning steps interact, including logical dependencies, incorporation of prior knowledge, and revisitation of earlier conclusions.

3.5 Graph Construction and Merging

The final reasoning graph combines the segmented nodes and annotated edges. A trace often states the same intermediate result more than once, so the same content would appear as several nodes and inflate the graph. We therefore compress repeated steps before building the graph. A *repetition* edge defines semantic equivalence: its endpoints

express the same content or a close rephrasing. Union-Find merges connected repetition components into equivalence classes. Node features are recomputed on the compressed graph G^* , and edge features come from the retained compressed edges. When several edges collapse onto the same pair we keep the strongest one; equal-score ties use the fixed ordering background $\prec$ observation $\prec$ meta-reflection $\prec$ inference. Merging collapses repeated reasoning steps while preserving external dependencies; because equivalence classes can redirect formerly forward edges across merged components, the resulting graph can contain cycles that capture revisitation and verification behavior. Appx. C.2 quantifies merging and tie resolution: 10.8% of nodes participate in repetition components, and 75.1% of equal-strength conflicts are same-type redirects, so the ordering is trivial in most cases.

4 Graph Modeling and Interpretation

We obtain the labels that supervise training (§4.1), train a GNN that predicts correctness from a CoT graph (§4.2), and use GNNExplainer for interpretability, attributing predictions back to nodes to identify which structures matter (§4.3).

4.1 Step-Level Supervision Labels

We use GPT-5 Mini (Singh et al., 2026) as an LLM judge to obtain step- and trace-level labels along three dimensions: (i) *Relevance* categorizes whether a step advances the reasoning process (*relevant*, *redundant*, or *irrelevant*). (ii) *Hallucination* evaluates the internal validity of each step in isolation (*correct*, *partial*, or *hallucinated*). (iii) *Final-answer correctness* is a correct/incorrect label. These labels serve as (i) supervision signals for the GNN models (§5.2), (ii) pruning labels for the efficiency analysis in §5.5, and (iii) evaluation labels for Best-of- N selection in §5.6. Thus, GPT-5 Mini defines target labels and Llama defines graph edges, but the correctness GNN receives only structural features and typed dependencies, not node text or judge labels. The experiment asks whether structure alone is sufficient to predict LLM-failure phenomena: a positive result indicates that the signal is encoded in reasoning topology, while a negative result means the signal may depend on node semantics, model or data scale, or may not be expressible from the graph alone. Appx. C.3 reports judge-ensemble agreement of 85.3% and Cohen $\kappa = 0.82$ over 95 edges; Appx. I lists all prompts.

4.2 GNN Modeling

The model is given only structure: node features and typed edges, with no step text. This isolates CoT structure from the semantic content of the steps, so whatever the model predicts must come from how the reasoning is organized. Given a compressed graph G^* , a GNN (Xu et al., 2019) maps node features and typed edges to prediction scores. Let $h_i^{(0)} = \phi(v_i)$ and let $\tilde{\mathcal{N}}^-(i) = \{i\} \cup \{j : (v_j, v_i) \in E\}$. The graph-level GATv2 encoder (Brody et al., 2022) updates node states as

$$h_i^{(\ell+1)} = \sigma \left(\sum_{j \in \tilde{\mathcal{N}}^-(i)} \alpha_{ji}^{(\ell)} W^{(\ell)} h_j^{(\ell)} \right),$$

$$\alpha_{ji}^{(\ell)} = \text{softmax}_{j \in \tilde{\mathcal{N}}^-(i)} a_{\theta}^{(\ell)}(h_j^{(\ell)}, h_i^{(\ell)}, \bar{\psi}_{ji}). \quad (1)$$

Here $a_{\theta}^{(\ell)}$ is the GATv2 attention score (we use GATv2 as a strong standard choice for graph classification), $\alpha_{ji}^{(\ell)}$ is normalized over the incoming neighborhood, and $\bar{\psi}_{ji}$ is the edge feature described above, with a self-loop feature when $j = i$. After $K = 2$ layers, mean pooling and an MLP give the outcome distribution

$$z_{G^*} = \frac{1}{|V|} \sum_{i \in V} h_i^{(K)}, \quad (2)$$

$$p_{G^*} = \text{softmax}(\text{MLP}(z_{G^*})) = f_{\theta}(G^*).$$

For node-level analysis, a two-layer GINE encoder (Hu et al., 2020) uses the same graph representation with two heads, producing p_i^{rel} for relevance and p_i^{hall} for hallucination at each step v_i . These judge labels are supervision targets, not inputs to the main correctness classifier. The predictor is permutation-equivariant before pooling and permutation-invariant after pooling; Appx. D compares it with graph-statistic, MLP, sequence, and feature/edge ablations.

4.3 Which Steps Carry the Outcome Signal

To identify recurring patterns in CoT structure, we need to know which parts of a graph a prediction relies on. GNNExplainer (Ying et al., 2019) answers this by learning a soft mask over nodes and edges that preserves the prediction while keeping the mask small. We apply it to the trained predictor f_{θ} , keeping its weights fixed. It optimizes a continuous node mask $m \in [0, 1]^{|V|}$ that preserves the original prediction while keeping the explanation

sparse and interpretable:

$$m^* = \arg \min_m \mathcal{L}(f_\theta(G^* \odot m), \hat{y}_{G^*}) + \lambda_1 \|m\|_1 + \lambda_2 H(m). \quad (3)$$

Here $\hat{y}_{G^*} = \arg \max_c p_{G^*}(c)$, $\mathcal{L}$ is cross-entropy with target $\hat{y}_{G^*}$, and $H(m)$ is mask entropy. This regularized objective approximates maximizing mutual information $\text{MI}(Y, G^* \odot m)$, where Y is the GNN prediction random variable. We use node masks only, so m_i^* scores the importance of step v_i ; since these scores are not on a common scale across graphs, we normalize each by its mean mask:

$$\tilde{m}_i = \frac{m_i^*}{|V|^{-1} \sum_{u \in V} m_u^*}. \quad (4)$$

Thus $\tilde{m}_i > 1$ denotes above-average importance within the same trace. These attributions explain the GNN predictor, not the LLM; we aggregate them by position, degree balance, and predicted outcome. Implementation details are in Appx. A.

5 Results and Discussion

We evaluate whether CoT structure provides information about reasoning quality. The central question is not only whether a trace is correct, but which structural properties of the CoT are related to correctness, redundancy, hallucination, and useful verification. Specifically, our experiments address five questions: whether graph structure predicts final-answer correctness better than LR and LCoT2Tree (§5.2); which step-level phenomena are structurally visible (§5.3); what GNNExplainer reveals about successful and failed reasoning (§5.4); how much long-CoT redundancy is useful verification rather than path inefficiency (§5.5); and whether structural signals improve Best-of- N selection (§5.6).

5.1 Experimental Setup

Evaluation Datasets and LLMs. Primary experiments use GPQA (Rein et al., 2024), a graduate-level science benchmark requiring multi-step reasoning (physics, chemistry, and biology) and 600 questions from MMLU-Pro (Wang et al., 2024b). We evaluate three open-weight reasoning models spanning controlled variation in scale and post-training strategy: DS-Qw7B and DS-Qw14B (Guo et al., 2025), trained via knowledge distillation from DeepSeek-R1, and MiMo-7B-RL (Xiaomi LLM-Core Team, 2025), trained with direct reinforcement learning on verifiable rewards. Appx. F

Model	LR	LCoT2Tree (Jiang et al., 2025b)	CoT2Graph (ours)
DS-Qw7B	65.9	68.2 (+2.3)	70.5 (+4.6)
DS-Qw14B	57.6	62.9 (+5.3)	65.4 (+7.8)
MiMo-7B-RL	61.4	63.0 (+1.6)	64.8 (+3.4)

Table 1: Final-answer correctness prediction accuracy (%) on GPQA. Parenthetical values indicate improvement over the Logistic Regression (LR) baseline. Our CoT2Graph consistently outperforms LCoT2Tree.

summarizes post-training differences among DS-Qw7B, DS-Qw14B, and MiMo-7B-RL, and reports the transfer setting used to test whether graph signals generalize across generators.

5.2 Structure Encodes Answer Correctness

Question. Does the structure of a CoT dependency graph carry predictive signal for final-answer correctness?

Finding. CoT2Graph consistently improves correctness prediction, outperforming LCoT2Tree by up to 2.5% and LR by up to 7.8%.

We frame the task as binary graph-level classification. Across all evaluation settings, CoT2Graph consistently outperforms LCoT2Tree, achieving gains of up to +2.5% (see Table 1). The advantage over LCoT2Tree is visible across all model families, demonstrating that the richer representational capacity of graphs (which can encode cycles, cross-step dependencies, and non-hierarchical chains) translates into predictive gains. We observe the largest margin over LCoT2Tree for DS-Qw14B (+2.5%), suggesting that a richer graph representation better captures its more complex reasoning.

A logistic regression on node count, edge count, density, and degree statistics performs below the length-only baseline. This suggests that the gain is not due only to graph size or density. Appx. D reports graph-statistic baselines, pooled and sequence encoders, and edge-type/node-feature ablations.

5.3 Structure Reveals Redundancy but Not Hallucination

Question. Which CoT step-level properties are structural?

Finding. Graph structure predicts whether a step is redundant, but carries little signal about hallucination, suggesting that intra-step and semantic failures require complementary signals (see §5.6).

Beyond final-answer prediction, we investigate whether graph topology encodes information about the quality of individual reasoning steps, and which structural properties the GNN uses to distinguish correct from incorrect CoT traces.

Relevance is an Inter-Step Structural Property.

Our node-level GNN trained to classify *relevant*, *redundant*, or *irrelevant* steps achieves 65.2% class accuracy on relevant nodes and 66.5% on redundant nodes (see Table 8), indicating that graph structure encodes a signal for detecting redundancy. Irrelevant nodes account for less than 1% of the dataset and are grouped with redundant nodes.

Hallucination is an Intra-Step Phenomenon.

The graphs that predict final-answer correctness and redundancy carry almost no signal about hallucinated² steps. Class accuracy falls to 7.0% on partially hallucinated nodes and 7.9% on hallucinated ones (Appx. H), against 65.2% and 66.5% for relevance under the same architecture. We read this gap as a finding rather than a shortcoming: it marks where dependency structure stops being informative. Relevance is a property of how a step relates to its neighbors and is therefore visible in the topology. In contrast, hallucination is primarily an intra-step semantic property and is therefore poorly captured by our dependency representation. The separation is also exploitable. Our CoT2Graph correlates weakly with entropy-based methods ($\rho = -0.16$), and the ensemble gains 1.6 points in Best-of- N (§5.6): structural and semantic verifiers should be ensembled.

5.4 Indicative Structural Patterns within CoT

Question. CoT structure encodes predictive signal, but which structural patterns separate incorrect/correct CoTs?

Finding. Correct CoTs rely on early planning; incorrect CoTs contain synthesis sinks merging many subprocesses.

GNNExplainer (§4.3) reveals two complementary structural patterns. First, correct CoTs place greater predictive importance on early planning and task-decomposition steps. Second, incorrect CoTs place greater importance on intermediate, high-in-degree synthesis nodes that combine multiple reasoning threads.

Early Planning Steps Predict Correctness. In true positive predictions, node importance concentrates on the first 20–25% of the steps (Figure 3). Manual inspection of these early nodes shows that they primarily contain problem formalization and planning steps, including decomposing the task into sub-problems and deciding the overall reasoning strategy. In true negative predictions, impor-

²Appx. I defines a hallucination as an error within a step, such as an incorrect computation or unsupported claim, rather than an error in its dependencies.

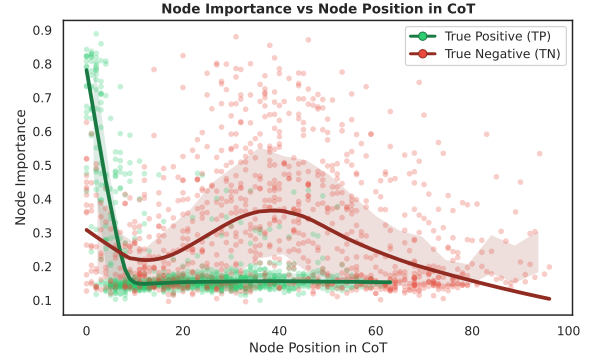


Figure 3: GNNExplainer node importance across CoT node positions (GPQA, DS-Qw7B). Early nodes are more important in correct CoTs, whereas intermediate nodes are more important in incorrect CoTs.

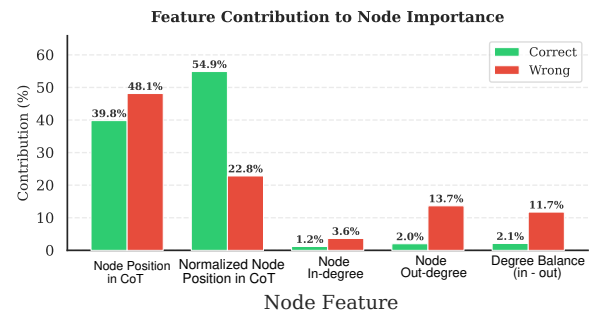


Figure 4: Linear regression of node-feature importance scores, comparing correct and incorrect CoTs.

tance shifts toward the middle of the trace. This pattern associates incorrect outcomes with intermediate reasoning structure; it does not establish that those nodes cause the error.

Planning and Synthesis Steps Are Crucial. We next examine how node connectivity (in-degree and out-degree)³ relates to final-answer correctness (see Figure 11). High out-degree nodes (*planner*) are typically early and correspond to planning and task decomposition in correct CoTs. High in-degree nodes (*synthesis sinks*) are more associated with incorrect CoTs and correspond to steps where the LLM conditions the generation on multiple previous reasoning threads, background information, and verification loops. Overall, correct reasoning shows initial planning and more distributed aggregation, while incorrect CoTs are characterized by dense synthesis steps.

Feature Attribution via Regression Analysis. A regression over GNNExplainer scores shows the same pattern: positional features dominate at-

³In-degree and out-degree refer, respectively, to the number of incoming and outgoing edges of a node in a graph.

tribution for correct CoTs, whereas connectivity features receive relatively greater importance for incorrect traces (Figure 4). Normalized position accounts for 54.9% of feature importance in correct CoTs and 22.8% in incorrect CoTs, concentrating attribution on early planning in successful traces. Absolute position receives more importance in incorrect CoTs, which are typically longer (Wu et al., 2026). Connectivity is more important in incorrect traces and is associated with subtask aggregation.

5.5 Reasoning Efficiency Analysis

Question. Do long CoTs reflect useful exploration and verification, or structural inefficiency?

Finding. Structural inefficiency dominates: it accounts for 49.0% of steps on average, compared with 6.6% for exploration/verification. Useful exploration emerges primarily in long CoTs (19.2%).

Structural inefficiency dominates CoT redundancy. Most long-CoT redundancy reflects structural inefficiency rather than useful verification. Across models, path inefficiency accounts for 49.0% of steps on average, compared with only 6.6% attributable to verification/exploration. Useful redundancy is concentrated in long traces, reaching 19.2%, whereas inefficiency is already present in short traces and increases with length (Table 2). To distinguish useful exploration (Wang et al., 2023) from unnecessary redundancy, we decompose each trace into two sources of efficiency loss: (i) exploration/verification, where the model revisits or repeats reasoning, and (ii) path inefficiency, where reasoning steps are removable or generated in a suboptimal dependency order. GPT-5 Mini relevance labels (§4.1) identify removable steps, while graph edges define dependency constraints.

Structural Efficiency Decomposition. We quantify two sources of efficiency loss (Figure 5). Let L_{CoT} be the original number of reasoning steps in the CoT, L_{loops} the trace length after collapsing repeated steps, and L_{TSP^*} the length of an approximate minimal traversal that respects the inferred graph dependencies. We then decompose total efficiency loss $\mathcal{L}$ into exploration/verification steps, $\mathcal{L}_{\text{verif.}}$, and path inefficiency, $\mathcal{L}_{\text{ineff.}}$:

$$\mathcal{L} = \underbrace{\frac{L_{\text{CoT}} - L_{\text{loops}}}{L_{\text{CoT}}}}_{\mathcal{L}_{\text{verif.}}} + \underbrace{\frac{L_{\text{loops}} - L_{\text{TSP}^*}}{L_{\text{CoT}}}}_{\mathcal{L}_{\text{ineff.}}} \quad (5)$$

$\mathcal{L}_{\text{verif.}}$ measures the steps removed by collapsing repetition loops, whereas $\mathcal{L}_{\text{ineff.}}$ measures the addi-

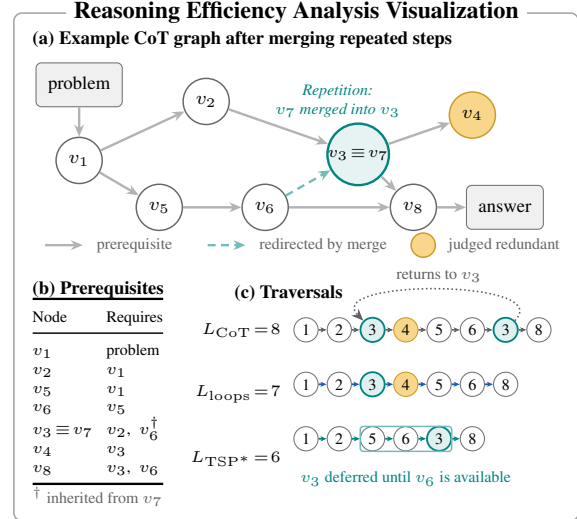


Figure 5: Example of the efficiency decomposition. v_7 repeats v_3 , so the two collapse into one node that inherits v_7 ’s prerequisite v_6 ; v_4 is judged redundant. L_{TSP^*} visits v_5 and v_6 before v_3 , due to node prerequisites.

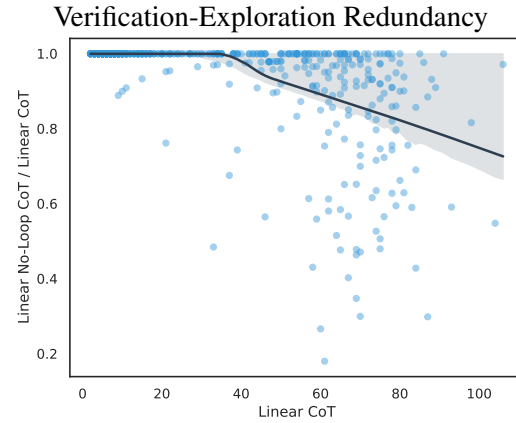


Figure 6: Scatter plot of original CoT length (L_{CoT} , x-axis) vs. efficiency ($E = 1 - \mathcal{L}_{\text{verif.}}$, y-axis), where $\mathcal{L}_{\text{verif.}}$ includes only good redundancy (verification/exploration) and excludes path inefficiency. We observe that verification loops start appearing once CoTs reach 35–45 steps, and then efficiency decreases approximately linearly as CoT length increases.

tional reduction obtained by following a minimal dependency-consistent traversal.

Graph-Based Optimal Path Construction. L_{TSP^*} is the length of a traversal that visits a step only once all its prerequisites are available. When no admissible next node exists, it relaxes the lowest-strength incoming prerequisite; equal-strength ties use the empirically derived type order. We use these prerequisite-relaxation events as the step-skipping signal for the hallucination analysis. Full details are given in Appx. G.

Model	Short ($p0-33$)		Medium ($p33-66$)		Long ($p66-100$)		Overall	
	$\mathcal{L}_{\text{verif.}}$	$\mathcal{L}_{\text{ineff.}}$	$\mathcal{L}_{\text{verif.}}$	$\mathcal{L}_{\text{ineff.}}$	$\mathcal{L}_{\text{verif.}}$	$\mathcal{L}_{\text{ineff.}}$	$\mathcal{L}_{\text{verif.}}$	$\mathcal{L}_{\text{ineff.}}$
DS-Qw7B	0.0	25.8	2.2	53.0	20.4	78.5	7.3	52.6
DS-Qw14B	0.0	9.7	0.0	34.1	18.0	61.8	5.5	35.5
MiMo-7B-RL	0.0	39.8	2.2	59.0	19.2	77.4	6.9	58.9
Average	0.0	25.1	1.5	48.7	19.2	72.6	6.6	49.0

Table 2: Decomposition of reasoning redundancy across CoT length on GPQA. $\mathcal{L}_{\text{verif.}}$ = good redundancy (exploration/verification); $\mathcal{L}_{\text{ineff.}}$ = bad redundancy (suboptimal order). Short/Medium/Long refer to CoT length quantile.

Empirical Findings. Across models, path inefficiency accounts for 49.0% of steps on average, against only 6.6% for exploration/verification, and Table 2 shows a clear length-dependent pattern. Exploration/verification redundancy is negligible in short traces and appears primarily after approximately 35–45 steps (Figure 6), rising to 19.2% in the longest traces. Path inefficiency, in contrast, is already present in short traces and rises across all length quantiles. Among the evaluated models, DS-Qw14B exhibits substantially lower path inefficiency (35.5%) than DS-Qw7B (52.6%) and MiMo-7B-RL (58.9%), despite comparable levels of verification redundancy.

Dependency Violations Are Associated with Step Errors. Complex reasoning often requires decomposing out-of-distribution problems into intermediate sub-tasks within the model’s capabilities, followed by composition of partial solutions. In this setting, skipping intermediate steps can be a valid shortcut but may also lead to hallucinations. We next test whether generating a step before its inferred prerequisites are satisfied is associated with node-level errors. We define a skip as a prerequisite-relaxation event during the L_{TSP} traversal. Relative to correct nodes, partially hallucinated nodes exhibit +6.96% higher skipping and hallucinated nodes +7.03%, consistently across models (Figure 10).

5.6 CoT2Graph Helps Best-of- N Decoding

Question. Can structural signals improve Best-of- N selection, and are they complementary to uncertainty?

Finding. CoT2Graph is the strongest standalone verifier; combining it with entropy improves accuracy to 56.1%.

We evaluate CoT2Graph as a *structural verifier* in Best-of- N sampling (Deng et al., 2024). Given N independent CoTs for the same question, we select the one most likely to yield a correct answer. Beyond absolute performance, we assess whether the methods capture complementary signals.

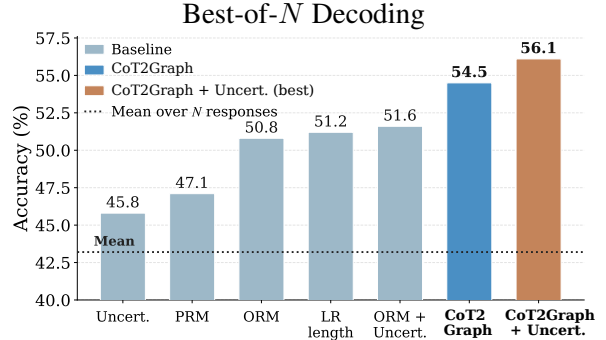


Figure 7: Best-of- N ($N = 5$) selection accuracy on MMLU-Pro with DS-Qw7B. CoT2Graph-Best achieves the highest performance among single verifiers (54.5%) with a baseline of 43.2%, outperforming all previous methods by at least 2.9%. Combining CoT2Graph with entropy selection further improves accuracy to 56.1%.

CoT Verifier Methods. We compare six selection strategies. The entropy method selects the CoT with the lowest mean per-token entropy (Fu et al., 2025). PRM-Best (Mistral-7B-PRM) aggregates step-level rewards (Lightman et al., 2024; Wang et al., 2024a), while ORM-Best (Skywork-Reward-Llama-8B) uses a final outcome score (Liu et al., 2026). Logistic regression ranks CoTs by length, and CoT2Graph by GNN-predicted correctness. We also report ORM combined with entropy (ORM + Uncert.), the strongest non-graph baseline. The PRM, ORM, and edge annotator have 7–8B parameters, but this does not make their costs or inputs equivalent: PRM and ORM score trace text, whereas the GNN receives graph-encoded structural features after edge annotation. Edge annotation considers at most $n(n-1)/2$ node pairs per trace.⁴ We evaluate on 600 MMLU-Pro questions using DS-Qw7B and $N=5$ samples. The mean baseline achieves 43.2%, while the oracle upper bound is 68.3%. CoT2Graph-Best reaches 54.5%, outperforming ORM-Best (50.8%) and PRM-Best (47.1%); the gain over the strongest non-graph verifier is 2.9 points (Figure 7).

⁴See Appx. E for details on runtime.

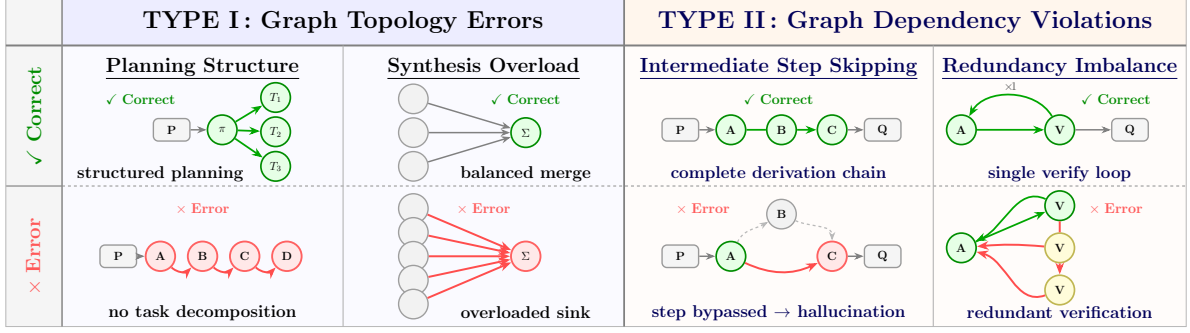


Figure 8: Structural reasoning patterns identified by the proposed **CoT2Graph** taxonomy, including planning structures, synthesis node overload, intermediate step skipping, and verification redundancy.

Orthogonal Signals and Ensemble Strategy.

CoT2Graph is weakly correlated with entropy-based selection ($\rho = -0.16$), but moderately correlated with PRM, ORM, and length-based ranking ($\rho > 0.4$) (see Figure 12). This suggests that topology and uncertainty capture complementary signals: combining CoT2Graph with entropy increases accuracy from 54.5% to 56.1%, the best overall result.

6 Structural CoT Error Taxonomy

Existing CoT evaluation frameworks, such as ROSCOE (Golovneva et al., 2023), mainly evaluate semantic properties (fluency, coherence, factual consistency, and logical validity) at the step level. They do not describe global structural patterns in the CoT (Figure 8). We organize the observed associations into two categories:

Type I: Graph Topology Errors (§5.4). Topology patterns associated with reasoning outcome.

- **Planning structure.** Correct CoTs often contain early planning nodes with moderate out-degree, corresponding to task decomposition and branching. In contrast, incorrect CoTs more often follow linear chains without explicit planning.
- **Synthesis overload.** Correct reasoning tends to distribute synthesis operations across the trajectory, while incorrect CoTs contain high in-degree sink nodes that merge many dependencies at once. They mark reasoning failures and fragile synthesis steps.

Type II: Graph Dependency Violations (§5.5).

Patterns in which the observed step order conflicts with the inferred dependencies.

- **Intermediate step skipping.** Some CoTs bypass required intermediate derivations and produce

conclusions before prerequisite subgoals are established, which raises hallucination rates.

- **Redundancy imbalance.** Moderate verification and exploration occur in correct traces, whereas excessive branching and repetition are associated with structural inefficiency and failure.

We also observe cases of structurally flawed reasoning with correct final answers, including traces that initially follow incorrect reasoning paths but self-correct near the final step, as well as multiple-choice reasoning by exclusion close to guessing.

7 Conclusion

We introduced CoT2Graph to study how the structure of CoT traces relates to reasoning quality. Representing long CoTs as typed dependency graphs improves correctness prediction over tree- and length-based baselines and enables interpretable analysis of reasoning patterns. GNNExplainer reveals that correct CoTs emphasize early planning and task decomposition, whereas incorrect CoTs are associated with overloaded synthesis steps and dependency violations. We further find that much of long-CoT redundancy reflects structural inefficiency rather than useful exploration or verification, and that structural signals complement uncertainty-based Best-of- N selection. Overall, our findings suggest that reasoning quality is reflected not only in individual steps, but also in how those steps are organized and composed, motivating moving from sequence- and step-level evaluation toward structure-aware analysis of reasoning traces.

Limitations

In this work, we employ an LLM-as-Judge setup for relevance and hallucination annotations, which may inherit biases from the GPT-5 Mini evalua-

tor (Singh et al., 2026). While this provides flexible and scalable supervision, alternative annotation strategies could rely on smaller NLI models or task-specific classifiers for edge and node labeling.

The quality of the resulting graphs is also influenced by the accuracy of the underlying step segmentation process, which determines the granularity of CoT decomposition and therefore the structure of the constructed graph.

Edge labels are generated by Llama-3.1-8B-Instruct and remain subject to judge and prompt bias. Appx. C.3 reports the 95-edge human comparison, where judge–ensemble agreement is 85.3% and Cohen κ is 0.82. Pairwise annotation also limits its use on long traces.

Overall, the goal of this framework is to analyze and expose structural signals in Chain-of-Thought reasoning traces, rather than to define a single optimal annotation or construction paradigm.

Ethical considerations

We do not identify specific ethical risks associated with our work. The framework supports interpretability by describing reasoning patterns associated with failures and hallucinations.

We believe that improving the transparency and analysis of reasoning processes can contribute to the development of more reliable and safer AI systems, particularly as LLMs are increasingly deployed in autonomous and agentic settings.

Acknowledgments

The authors thank the reviewers for their constructive feedback. T. F. Banfi is currently affiliated with CERN; this work was conducted before, while he was at ETH Zurich (LRE Lab). M. Akhtar was supported by the ETH AI Center through an ETH AI Center postdoctoral fellowship.

We thank Stefano Marin (CERN) and Noah Julian Stäubli (CERN) for their thoughtful comments and feedback in the late stage of this work.

We also thank Francesco Dorati (Politecnico di Milano) for his help and time dedicated to the human validation and reliability analysis of the LLM-as-a-Judge edge annotations.

References

Tommaso Felice Banfi and Sashenka Gamage. 2026. [Llms for game theory: Entropy-guided in-context learning and adaptive cot reasoning](#). *Preprint*, arXiv:2601.10775.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefer. 2024. [Graph of thoughts: solving elaborate problems with large language models](#). In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI’24/IAAI’24/EAAI’24. AAAI Press.

Shaked Brody, Uri Alon, and Eran Yahav. 2022. [How attentive are graph attention networks?](#) In *International Conference on Learning Representations*.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. 2025. [Research: Learning to reason with search for LLMs via reinforcement learning](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. [Explaining answers with entailment trees](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shumin Deng, Ningyu Zhang, Nay Oo, and Bryan Hooi. 2024. [Towards a unified view of answer calibration for multi-step reasoning](#). In *Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ACL 2024)*, pages 25–38, Bangkok, Thailand. Association for Computational Linguistics.

Matthias Fey and Jan Eric Lenssen. 2019. [Fast graph representation learning with pytorch geometric](#). *Preprint*, arXiv:1903.02428.

Yichao Fu, Xuwei Wang, Yuandong Tian, and Jiawei Zhao. 2025. [Deep think with confidence](#). *Preprint*, arXiv:2508.15260.

Olga Golovneva, Moya Peng 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](#). In *The Eleventh International Conference on Learning Representations*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.

- 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.
- Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S Pande, and Jure Leskovec. 2020. [Strategies for pre-training graph neural networks](#). In *International Conference on Learning Representations*.
- Alon Jacovi, Yonatan Bitton, Bernd Bohnet, Jonathan Herzig, Or Honovich, Michael Tseng, Michael Collins, Roei Aharoni, and Mor Geva. 2024. A chain-of-thought is as strong as its weakest link: A benchmark for verifiers of reasoning chains. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4615–4634.
- Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanwei Li, Yu Qi, Xinyan Chen, Liuhui Wang, Jianhan Jin, Claire Guo, Shen Yan, Bo Zhang, Chaoyou Fu, Peng Gao, and Hongsheng Li. 2025a. [MME-CoT: Benchmarking chain-of-thought in large multimodal models for reasoning quality, robustness, and efficiency](#). In *Forty-second International Conference on Machine Learning*.
- Gangwei Jiang, Yahui Liu, Zhaoyi Li, Wei Bi, Fuzheng Zhang, Linqi Song, Ying Wei, and Defu Lian. 2025b. [What makes a good reasoning chain? uncovering structural patterns in long chain-of-thought reasoning](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6490–6514, Suzhou, China. Association for Computational Linguistics.
- Jinu Lee and Julia Hockenmaier. 2025. [Evaluating step-by-step reasoning traces: A survey](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 1789–1814, Suzhou, China. Association for Computational Linguistics.
- Xiaoyuan Li, Wenjie Wang, Moxin Li, Junrong Guo, Yang Zhang, and Fuli Feng. 2024. [Evaluating mathematical reasoning of large language models: A focus on error identification and correction](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11316–11360, Bangkok, Thailand. Association for Computational Linguistics.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *The Twelfth International Conference on Learning Representations*.
- Jiaen Lin, Jingyu Liu, and Yingbo Liu. 2025. [Optimizing multi-hop document retrieval through intermediate representations](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15798–15809, Vienna, Austria. Association for Computational Linguistics.
- Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiakai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. 2026. [Skywork-reward-v2: Scaling preference data curation via human-ai synergy](#). *Preprint*, arXiv:2507.01352.
- Venkatesh Mishra, Bimsara Pathiraja, Mihir Parmar, Sat Chidananda, Jayanth Srinivasa, Gaowen Liu, Ali Payani, and Chitta Baral. 2025. [Investigating the shortcomings of LLMs in step-by-step legal reasoning](#). In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7810–7841, Albuquerque, New Mexico. Association for Computational Linguistics.
- Shentong Mo and Miao Xin. 2024. [Tree of uncertain thoughts reasoning for large language models](#). In *ICASSP 2024 – 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12742–12746.
- Danilo Neves Ribeiro, Shen Wang, Xiaofei Ma, Rui Dong, Xiaokai Wei, Henghui Zhu, Xinchu Chen, Peng Xu, Zhiheng Huang, Andrew Arnold, and Dan Roth. 2022. [Entailment tree explanations via iterative retrieval-generation reasoner](#). In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 465–475, Seattle, United States. Association for Computational Linguistics.
- Jingwei Ni, Ekaterina Fadeeva, Tianyi Wu, Mubashara Akhtar, Jiaheng Zhang, Elliott Ash, Markus Leippold, Timothy Baldwin, See-Kiong Ng, Artem Shelmanov, and Mrinmaya Sachan. 2026. [Reprobe: Efficient test-time scaling of multi-step reasoning by probing internal states of large language models](#). *Preprint*, arXiv:2511.06209.
- Archiki Prasad, Swarnadeep Saha, Xiang Zhou, and Mohit Bansal. 2023. [ReCEval: Evaluating reasoning chains via correctness and informativeness](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10066–10086, Singapore. Association for Computational Linguistics.
- Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. [GPQA: A graduate-level google-proof Q&A benchmark](#). In *First Conference on Language Modeling*.
- 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.
- Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, and 467 others. 2026. [Openai gpt-5 system card](#). *Preprint*, arXiv:2601.03267.
- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024a. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. [Self-consistency improves chain of thought reasoning in language models](#). In *The Eleventh International Conference on Learning Representations*.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. 2024b. [Mmlu-pro: A more robust and challenging multi-task language understanding benchmark](#). In *NeurIPS*.
- Jiayi Weng. 2026. [Learning beyond gradients](#). Blog post, <https://trinkle23897.github.io/learning-beyond-gradients/>, accessed September 2026.
- Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2026. [When more is less: Understanding chain-of-thought length in LLMs](#). In *The Fourteenth International Conference on Learning Representations*.
- Xiaomi LLM-Core Team. 2025. [MiMo: Unlocking the reasoning potential of language model – from pre-training to posttraining](#). *Preprint*, arXiv:2505.07608.
- Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. [How powerful are graph neural networks?](#) In *International Conference on Learning Representations*.
- Xin Xu, Shizhe Diao, Can Yang, and Yang Wang. 2025. [Can we verify step by step for incorrect answer detection?](#) In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 8322–8330. International Joint Conferences on Artificial Intelligence Organization. Main Track.
- Shangzi Xue, Zhenya Huang, Jiayu Liu, Xin lin, Yuting Ning, Binbin Jin, Xin Li, and Qi Liu. 2024. [Decompose, analyze and rethink: Solving intricate problems with human-like reasoning cycle](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 357–385. Curran Associates, Inc.
- Tianci Xue, Ziqi Wang, Zhenhailong Wang, Chi Han, Pengfei Yu, and Heng Ji. 2023. [Rcot: Detecting and rectifying factual inconsistency in reasoning by reversing chain-of-thought](#). *Preprint*, arXiv:2305.11499.
- 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. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23*, Red Hook, NY, USA. Curran Associates Inc.
- Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: generating explanations for graph neural networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA. Curran Associates Inc.
- Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. 2023. [Verify-and-edit: A knowledge-enhanced chain-of-thought framework](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5823–5840, Toronto, Canada. Association for Computational Linguistics.

Contents		E Computational Scaling Analysis	16
1 Introduction	1	F Cross-Model Generalization	17
2 Related Work	2	F.1 Model Characterization	17
3 CoT2Graph Framework	2	F.2 Cross-Model Transfer Results . .	18
3.1 Generation of CoT Traces	3	G Optimal TSP* Path Details	18
3.2 Graph Representation	3	H Additional Figures and Tables	19
3.3 Node Segmentation	3	I Prompts	20
3.4 Edge Annotation	3		
3.5 Graph Construction and Merging .	3		
4 Graph Modeling and Interpretation	4		
4.1 Step-Level Supervision Labels . .	4		
4.2 GNN Modeling	4		
4.3 Which Steps Carry the Outcome Signal	4		
5 Results and Discussion	5		
5.1 Experimental Setup	5		
5.2 Structure Encodes Answer Correct- ness	5		
5.3 Structure Reveals Redundancy but Not Hallucination	5		
5.4 Indicative Structural Patterns within CoT	6		
5.5 Reasoning Efficiency Analysis . .	7		
5.6 CoT2Graph Helps Best-of- N De- coding	8		
6 Structural CoT Error Taxonomy	9		
7 Conclusion	9		
A CoT2Graph Implementation Details	14		
A.1 LLM Generation Hyperparameters	14		
A.2 Node Segmentation	14		
A.3 Node Feature Vector	14		
A.4 Edge Classification	14		
A.5 Graph Compression via Union-Find	14		
A.6 GNN Architecture and Training .	14		
B CoT Segmentation Analysis	15		
B.1 Rule-Based Segmentation Sensitivity	15		
B.2 Embedding-Based Segmentation .	15		
C Edge Type Definitions and Validation	15		
C.1 Definitions and Literature Support	15		
C.2 Semantic Equivalence and Conflict Resolution	15		
C.3 Human Annotation Validation . .	16		
D Ablation Studies	16		

A CoT2Graph Implementation Details

This appendix provides implementation details of the CoT2Graph framework, covering the full process from LLM hyperparameters to the graph construction pipeline and GNN configuration.

A.1 LLM Generation Hyperparameters

All experiments share the same generation configuration across all models (DS-Qw7B, DS-Qw14B, MiMo-7B-RL) and datasets (GPQA, MMLU-Pro) to ensure comparability. Inference was performed locally using the vLLM engine and weights were loaded in bfloat16 precision. Experiments were run on NVIDIA A100, RTX 3090 and 4090 GPUs. We use a low-temperature decoding setup with temperature $T = 0.1$, a maximum generation length of $2^{14} \approx 16K$ tokens, batch size of 16, and fixed random seed 42. These hyperparameters are chosen to favour stable outputs while preserving stochasticity for diversity in Best-of- N evaluation. During generation, we also collect the top- $K = 5$ token log-probabilities to compute uncertainty metrics.

A.2 Node Segmentation

Raw CoT text is segmented into atomic steps using a keyword-based parser. Transition markers (*Step*, *Wait*, *So*, *Finally*, *Let me verify*, *However*, *Therefore*, and related phrases) are identified via case-insensitive regex matching similar to previous work (Jiang et al., 2025b). Segments shorter than 100 characters are merged with the following step to ensure semantic completeness. The resulting sequence $v_1, v_2, \dots, v_n$ preserves linear generation order. Parser robustness and the embedding-based alternative are described in Appx. B.

A.3 Node Feature Vector

Let $n = |\mathcal{V}|$. Each node v_i is represented by a six-dimensional feature vector (Jiang et al., 2025b):

$$\phi(v_i) = [i, i/n, d_i^{\text{in}}/n, d_i^{\text{out}}/n, d_i^{\text{in}} - d_i^{\text{out}}, d_i^{\text{in}}/(d_i^{\text{out}} + 1)],$$

encoding absolute and normalized position, in-/out-degree, and degree balance and ratio.

A.4 Edge Classification

Dependency relationships are inferred by Llama-3.1-8B-Instruct (Grattafiori et al., 2024) applied to the upper triangular half of the adjacency matrix: for all pairs (v_i, v_j) with $i < j$, the model returns

an edge type $t \in \mathcal{T}$ and a strength $s \in [0, 10]$. Restricting to the upper triangle reduces prompt cost and encodes a causal prior consistent with sequential generation of the LLM. Appx. I includes the prompt used for edge classification.

The six edge types are: *inference* (strict logical or mathematical deduction from v_i to v_j), *background* (prior knowledge from v_i contextualizing v_j), *observation* (direct empirical support from v_i for v_j), *meta-reflection* (reasoning about v_i 's validity that informs v_j), *repetition* (v_i and v_j express the same content or a semantic rephrasing), and *none* (no meaningful dependency).

A.5 Graph Compression via Union-Find

Nodes connected by *repetition* edges are merged with Union-Find. Each connected component becomes a canonical supernode, and its incident edges are redirected through the node-to-representative map. This removes repeated formulations while preserving their dependencies.

A.6 GNN Architecture and Training

Implementation uses PyTorch Geometric (Fey and Lenssen, 2019).

Graph-Level Outcome Prediction. The GATv2 (Brody et al., 2022) classifier comprises two GATv2Conv layers (hidden dimension 64, single attention head, edge features incorporated via attention weighting), global mean pooling, and a two-layer MLP head (Linear(64 $\rightarrow$ 64)–ReLU–Dropout(0.2)–Linear(64 $\rightarrow$ 2)). Training uses Adam (lr = 10^{-3} , weight decay 10^{-4}), batch size 32, StepLR scheduling (step 20, $\gamma = 0.5$), early stopping (patience 50), and 200 maximum epochs.

Node-Level Classifiers. The GINE (Hu et al., 2020) classifier uses a two-layer encoder (hidden dimension 128) with multi-task prediction heads, cosine annealing scheduling, and 300 maximum epochs. GINE's sum aggregation is preferred over attention-based pooling for node-level tasks because it provides stronger inductive bias for distinguishing local neighborhood configurations.

Training Protocol. Graph-level results use 11 trace-level splits with seed 11. For the 545 GPQA graphs, each split assigns 445 graphs to training and validation and 100 to testing; 15% of the former are held out for validation before training-set class balancing. Early stopping selects validation F1. Node-level experiments use 400 training and

145 test graphs over 11 splits, and Best-of- N a question-level 70/30 split with seed 42, reserving 20% of training questions for validation.

B CoT Segmentation Analysis

B.1 Rule-Based Segmentation Sensitivity

We use a regex parser to avoid an additional GPT-5 Mini call for every trace. For each model, its transition markers were obtained by combining human inspection with GPT-5 Mini keyword extraction on a subset of traces. Their consistency across questions and datasets may reflect shared post-training conventions, so the marker list is model-specific.

We parsed all 545 GPQA traces from DS-Qw7B, obtaining 62.1 nodes per trace on average, and judged them against the GPT-5 Mini reference segmentation. Across these traces, 91.5% of parser nodes matched, 8.4% were undersplit, and fewer than 0.1% were oversplit. At trace level, 79.3% had at most 5% incorrectly segmented nodes and 86.7% had at most 10%. The 100-character merge favors preserving context over fragmenting a coherent step: for example, markers in a short span of the form “FIRST ... SO ...” can denote one argument rather than two steps.

Coding tasks were not evaluated. On MATH500 the parser stays reliable because reasoning models pair every derivation with prose carrying the same markers. The DS-Qw7B step below opens with *Now* (in bold) and already exceeds the merge threshold, so the parser keeps it as one node.

MATH500_{ID 404} — DS-Qw7B

Now, determine the number of boxes with neither pens nor pencils. Subtract the number of boxes with pens or pencils from the total:
Neither = Total boxes - |P or Q|
Neither = 10 - 7 = 3

Thresholds and markers may still require dataset- and model-specific tuning.

B.2 Embedding-Based Segmentation

We considered semantic segmentation before adopting the regex parser. CoT segmentation is under-determined: adjacent sentences may be separate derivation steps, connective tissue, or fragments of one calculation. In pilot runs, embedding neighborhoods often tracked sentence length and recurring discourse markers more than reasoning content. We tested K-means on sentence embeddings, dynamic programming with a within-segment similarity objective and complexity penalty, multi-scale

boundary scoring with Ward-style merging, and an optimal-coverage objective over contiguous windows. We also tried gradient-free policy search in the sense of Weng (2026). None of these variants produced boundaries stable enough for the graph analyses, so all reported results use the rule-based parser.

C Edge Type Definitions and Validation

C.1 Definitions and Literature Support

The edge annotator assigns one of five relations, or *none*, to each ordered node pair A and B:

- **Inference:** Node A is an explicit premise from which Node B follows directly.
- **Background information:** Node A contains prior knowledge, context, assumptions, or general facts that are used in Node B.
- **Observation:** Node B is an observation, comment, or general step related to Node A, even indirectly.
- **Meta-reflection:** Node B plans, checks, or reflects on the reasoning process rather than advancing the solution.
- **Repetition:** Nodes A and B express the same or a closely similar idea, result, or reasoning step.
- **None:** A and B have no meaningful relation.

The annotator also assigns a strength score: 0 for no link; 1–3 for a weak relation; 4–6 for a moderate but nonessential contribution; 7–8 when Node A is a primary premise; and 9–10 when Node B follows necessarily from Node A alone.

The taxonomy combines inference and support relations from entailment trees (Dalvi et al., 2021; Neves Ribeiro et al., 2022), background knowledge from retrieval-based reasoning (Lin et al., 2025), and reflection and repetition from search-based reasoning (Yao et al., 2023; Mo and Xin, 2024; Jiang et al., 2025b). Graph of Thoughts motivates representing these relations as edges between thought nodes (Besta et al., 2024).

C.2 Semantic Equivalence and Conflict Resolution

Two nodes are semantically equivalent when they express the same intermediate conclusion or a rephrasing with the same meaning. The annotator marks this relation as *repetition*; Union-Find

merges each connected repetition component into a supernode and redirects its incident edges. Overall, 10.8% of nodes participate in such components.

Redirected edges can assign different types to the same pair of supernodes. We retain the edge with the highest LLM strength; equal-strength ties use inference > meta-reflection > observation > background. This order follows the mean annotated strength by type across the evaluated models and datasets and agrees with the GNNExplainer attribution ranking. Among tied-strength comparisons, 75.1% are trivial because both edges already have the same type, since they link to nodes with equivalent content.

C.3 Human Annotation Validation

We validate edge labels on three GPQA CoTs generated by DS-Qw14B, containing 60, 61, and 49 nodes (170 total). The evaluated edges are sampled with stratification across judge-assigned edge classes: few calibration examples shown with the judge’s label and 95 blind examples. Three annotators label each edge independently: one paper author, one external participant familiar with LLM evaluation, and Fable Claude. Their majority vote gives one ensemble label per edge.

The judge is one annotation under comparison, not a gold standard. Since sampling is stratified by its labels, the row totals of Table 4 are fixed by design and do not show how often each relation occurs. Table 3 shows that judge–ensemble agreement is 85.3% with Cohen’s $\kappa = 0.82$, while Table 4 compares the two class by class.

Human Reviewer Instructions

TASK: Follow and read the definitions below (see: Prompt 3 - Edge Classification). For each ordered node pair, select one relation type, assign confidence on a 1--5 scale, and optionally add a comment.

	Judge	Author	Extern.	Fable	Ens.
Judge (Llama-3.1-8B)	–	86.3%	81.1%	84.2%	85.3%
Author	0.83	–	84.2%	87.4%	94.7%
External	0.77	0.81	–	82.1%	89.5%
Fable Claude	0.80	0.84	0.78	–	92.6%
Ensemble	0.82	0.93	0.87	0.91	–

Table 3: Pairwise agreement between annotations: raw agreement above the diagonal, Cohen’s κ below.

Judge \ Ens.	Inf	Bkg	Obs	Meta	Rep	None	Recall
Inference	18	1	1	0	0	0	90.0%
Bkg. info	0	16	0	1	0	2	84.2%
Observation	0	1	17	2	0	1	81.0%
Meta-refl.	0	0	0	13	1	1	86.7%
Repetition	0	0	0	0	1	0	100.0%
None	0	1	1	1	0	16	84.2%
n	18	19	19	17	2	20	95
Precision	100.0%	84.2%	89.5%	76.5%	50.0%	80.0%	
F1	0.95	0.84	0.85	0.81	0.67	0.82	

Table 4: LLM-as-Judge against the annotator ensemble on the sampled edges. Cells are edge counts; recall and precision are the diagonal over the row and column totals. Accuracy 85.3%, Cohen’s $\kappa = 0.82$. *Repetition* is near-absent here because it concentrates in long traces (§5.5), which are out of scope and too costly to annotate.

D Ablation Studies

Table 5 separates model baselines from edge, node-feature, and topology ablations on the same fixed splits; each ablation removes one candidate signal source. Logistic regression and random forest use graph size, density, and degree summaries; the pooled MLP removes message passing; and the sequence encoder removes graph edges. The untyped GNN preserves connectivity and strength while replacing the edge-type vector with a constant.

The graph-statistics logistic regression performs below the length-only baseline, indicating that coarse summaries alone do not explain the full model’s gain. All variants are run on GPQA traces from DS-Qw7B over the same 11 fixed splits; reported values are accuracy means with standard deviations across splits. Edge types are masked one channel at a time: edges stay in the graph and reach the GNN with a neutral type, so connectivity is unchanged and only type signal is lost. Their contributions are additive only if the remaining channels are unaffected.

Removing the *inference* channel produces the largest single-type drop (63.0), followed by *background* (64.2) and *observation* (65.8), whereas masking *meta-reflection* changes the score least (67.6). Removing the degree features costs more than removing position (66.4 against 68.3).

E Computational Scaling Analysis

CoT2Graph is suitable for offline analysis of fixed traces, while online use is limited by pairwise edge annotation. Edge annotation dominates the cost of the pipeline. For a trace with n nodes it queries Llama-3.1-8B-Instruct on at most $n(n - 1)/2$ or-

Group	Method / variant	Controlled input change	Accuracy (%)	Δ vs. ours
Baseline	Graph-statistics RF	same summaries as LR	58.3 $\pm$ 4.0	-12.2
	Graph-statistics LR	size, density, degree summaries	60.5 $\pm$ 4.5	-10.0
	<u>Length-only LR</u>	CoT length only	<u>65.9</u> $\pm$ 3.1	-4.6
	Sequence encoder	ordered node features, no graph edges	66.1 $\pm$ 3.6	-4.4
	<u>LCoT2Tree (Jiang et al., 2025b)</u>	LLM-annotated tree sketch instead of the dependency graph	<u>68.2</u> $\pm$ 2.8	-2.3
	Pooled-feature MLP	pooled node/edge features, no message passing	68.4 $\pm$ 4.1	-2.1
Edge ablation	Without inference type	mask inference channel only	63.0 $\pm$ 3.4	-7.5
	Without background type	mask background channel only	64.2 $\pm$ 4.4	-6.3
	Without observation type	mask observation channel only	65.8 $\pm$ 4.6	-4.7
	Untyped-edge GNN	keep connectivity/strength; replace type one-hot by a constant	66.1 $\pm$ 4.8	-4.4
	Without repetition type	mask repetition channel only	67.5 $\pm$ 3.8	-3.0
	Without meta-reflection type	mask meta-reflection channel only	67.6 $\pm$ 5.1	-2.9
	Without edge strength	keep topology and edge types	68.6 $\pm$ 5.0	-1.9
Node ablation	Without degree features	remove in/out degree, balance, and ratio	66.4 $\pm$ 4.3	-4.1
	Without position	remove absolute/normalized position	68.3 $\pm$ 5.3	-2.2
Extension	Semantic node embeddings	add frozen sentence embeddings to structural features	64.8 $\pm$ 5.5	-5.7
Reference	CoT2Graph (GATv2 GNN)	graph topology + edge type + strength + node features	70.5 $\pm$ 2.3	0.0

Table 5: Controlled baselines and ablations for final-answer prediction on GPQA (DS-Qw7B), accuracy (%) over the same 11 fixed splits, mean $\pm$ s.d. Rows are sorted by increasing score within each group; Δ is against CoT2Graph. The **bold** row is our CoT2Graph; the two underlined rows are the length LR and LCoT2Tree baselines of Table 1. Abbrev.: LR (logistic regression), RF (random forest), MLP (multi-layer perceptron), GNN (graph neural network).

dered pairs. Pairs are batched and traces are distributed across GPUs, without assuming ideal linear speedup. On 545 GPQA DS-Qw7B traces with a mean of 62 nodes, annotation takes 2.28 minutes per trace on an RTX 3090; the corresponding A100 and H100 figures, 0.21 and 0.12 minutes, are extrapolations rather than measurements. A log-log regression of annotation time on trace length gives an exponent of 1.31, below the $O(n^2)$ worst case.

The remaining stages are negligible next to annotation. Graph compression and feature extraction are linear in the number of edges, and the GATv2 classifier has fewer than 15K parameters and trains on 545 graphs, so it takes negligible GPU time.

F Cross-Model Generalization

We ask whether topology learned from one generator transfers to another. A structurally grounded CoT representation should, in principle, capture reasoning patterns that transcend the idiosyncrasies of any particular model; the graph topology induced by correct vs. incorrect reasoning should share universal features regardless of which model produced the trace. We test this hypothesis through a cross-model transfer study, training the GNN classifier on traces from one model and evaluating on traces from another. Beyond validating our method, this analysis reveals how post-training and model scale shape the underlying reasoning structure.

F.1 Model Characterization

We evaluate three models chosen to span controlled variation across two key axes: *model scale* (7B vs. 14B) and *post-training strategy* (knowledge distillation vs. direct reinforcement learning). Table 6 summarizes their principal characteristics.

DS-Qw7B & 14B. Both models are initialized from the Qwen2.5 base models (Qwen et al., 2025) and trained exclusively via supervised fine-tuning (SFT) on approximately 800k reasoning traces distilled from the DeepSeek-R1 teacher model (Guo et al., 2025), itself trained with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) through a four-stage pipeline (cold-start SFT $\rightarrow$ RL $\rightarrow$ SFT $\rightarrow$ RL alignment). Critically, the DS-Qw7B and DS-Qw14B distillation models receive *no direct RL signal*; their reasoning behavior is shaped entirely by behavioral cloning from the teacher’s outputs.

MiMo-7B-RL. MiMo (Xiaomi LLM-Core Team, 2025) is quite distant from DeepSeek. Its pre-training curriculum spans three stages with multi-token prediction objectives, making reasoning performance a first-class training objective. Post-training applies direct reinforcement learning on 130k verifiable mathematics and programming problems, using difficulty-aware reward sampling.

Model	Size	Base	Post-training	Key Characteristics
DS-Qw7B	7B	Qwen2.5-Math-7B	SFT on distilled CoTs (no direct RL)	~800k traces from DeepSeek-R1 teacher
DS-Qw14B	14B	Qwen2.5-14B	SFT on distilled CoTs (no direct RL)	Same pipeline; larger capacity for long-context
MiMo-7B-RL	7B	MiMo (Xiaomi)	Direct RL with verifiable rewards	130k math/code problems; difficulty-aware sampling

Table 6: Summary of the three reasoning models evaluated in the cross-model generalization study. Models are selected to provide controlled variation in scale and post-training strategy.

Method (F1 score)	Train ($\downarrow$) / Test ($\rightarrow$)	DS-Qw7B	DS-Qw14B	MiMo-7B-RL
LCoT2Tree (Jiang et al., 2025b)	DS-Qw7B	59.4	56.5	46.6
	DS-Qw14B	57.8	56.1	42.6
	MiMo-7B-RL	56.6	52.3	42.9
CoT2Graph (<i>ours</i>)	DS-Qw7B	62.0 (+2.6)	56.6 (+0.1)	50.0 (+3.4)
	DS-Qw14B	57.3 (-0.5)	58.2 (+2.1)	45.8 (+3.2)
	MiMo-7B-RL	60.7 (+4.1)	54.1 (+1.8)	51.5 (+8.6)

Table 7: Cross-model generalization: macro F1 (%) for graph-level correctness prediction on GPQA. Rows indicate the LLM that generated the CoT traces used to train the GNN predictor model, while columns indicate the CoT traces used for testing. CoT2Graph improves over LCoT2Tree in most settings, with gains up to +8.6% F1.

F.2 Cross-Model Transfer Results

Table 7 reports F1 scores for graph-level correctness classification. We report macro F1 as the comparison metric in this analysis rather than accuracy because class imbalance varies across models, reflecting their differing baseline accuracy on GPQA.

Overall Generalization. CoT2Graph consistently outperforms LCoT2Tree in cross-model transfer. The most pronounced improvements occur on MiMo-7B-RL traces: within-model (51.5% vs. 42.9%, +8.6%) and in DS-Qw7B $\rightarrow$ MiMo-7B-RL transfer (50.0% vs. 46.6%, +3.4%). These results confirm that graph-based structural representations encode reasoning patterns that generalize more broadly across models than features derived from LLM-generated reasoning tree sketches. Moreover, LCoT2Tree requires a frontier model (e.g., DeepSeek-v3) to produce the intermediate sketch used for tree construction, introducing a costly and model-dependent preprocessing step. CoT2Graph, by contrast, uses only a fixed 7B edge classifier, a substantially more scalable and deployment-friendly architecture.

Same Model Family, Different Scale. Transfer within the DeepSeek distillation family is moderate and approximately symmetric: 57.3% (train DS-Qw14B, test DS-Qw7B) vs. 57.8% (train DS-Qw7B, test DS-Qw14B) for CoT2Graph. Both models are trained with the same distillation pipeline from the same teacher, so their reasoning graphs share the same high-level structural vocabulary. The slight asymmetry (training on DS-Qw7B

generalizes marginally better to DS-Qw14B than vice versa) may reflect that 7B traces, being somewhat simpler in structure, produce features that transfer across the capacity gap more robustly.

Different Model Family, Matched Scale. The most informative comparison is between the two 7B models, which share scale but diverge in post-training strategy. Transfer is notably asymmetric: training on MiMo-7B-RL and testing on DS-Qw7B yields 60.7%, while the reverse yields only 50.0%. This asymmetry reveals that RL-trained models produce structurally richer reasoning graphs whose patterns subsume those of distillation-trained models. A GNN trained on MiMo’s RL-driven traces (with greater topological diversity, more frequent exploration loops, and more varied synthesis node configurations) learns a generalizable structural vocabulary of correct and incorrect reasoning. Conversely, one trained on the more uniform distillation traces of DS-Qw7B transfers less well to MiMo-7B-RL’s broader structural patterns.

The Role of Post-Training Strategy. MiMo-7B-RL’s lower within-model F1 (51.5%) compared to DS-Qw7B (62.0%) and DS-Qw14B (58.2%) reflects the *higher structural diversity of RL-trained traces*: greater topological variance makes binary classification harder in absolute terms, but provides richer training signal for cross-model transfer.

G Optimal TSP* Path Details

We detail the path construction used to estimate L_{TSP^*} , covering the traversal and the relaxation ap-

plied when a circular dependency blocks progress. Algorithm 1 estimates L_{TSP^*} with a dependency-consistent traversal. A blocked traversal removes the edge with the lowest LLM strength. Strength is the primary criterion; only equal-strength ties use the removal order background > observation > meta-reflection > inference. The importance order follows the mean annotated strength by type across the evaluated models and datasets and is consistent with the GNNExplainer attribution ranking. Appx. C.2 reports the related tie analysis, and Figure 9 relates loop-free CoT length to $\mathcal{L}_{ineff.}$, the path inefficiency measured against L_{TSP^*} .

Algorithm 1: Graph-Based Optimal Path Construction (TSP-Inspired)

Input: CoT graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; LLM-as-Judge relevance labels per node
Output: Ordered path $\mathcal{P}$; path length L_{TSP^*} ; relaxation count τ

- 1 Remove all nodes labelled *redundant* or *irrelevant*
- 2 $\mathcal{S} \leftarrow \{v_0\}$ // visited; v_0 = problem node
- 3 $\mathcal{R} \leftarrow \mathcal{V} \setminus \{v_0\}$; $\mathcal{P} \leftarrow [v_0]$; $\tau \leftarrow 0$
- 4 **while** $\mathcal{R} \neq \emptyset$ **do**
- 5 $\mathcal{F} \leftarrow \{v \in \mathcal{R} \mid \forall (u, v) \in \mathcal{E} : u \in \mathcal{S}\}$
 // prerequisites satisfied
- 6 **if** $\mathcal{F} \neq \emptyset$ **then**
- 7 $v^* \leftarrow \arg \min_{v \in \mathcal{F}} \text{id}(v)$
- 8 Append v^* to $\mathcal{P}$; $\mathcal{S} \leftarrow \mathcal{S} \cup \{v^*\}$;
 $\mathcal{R} \leftarrow \mathcal{R} \setminus \{v^*\}$
- 9 **else**
- 10 // deadlock: relax prereq.
 Remove the minimum-strength blocking edge (u, v) with $v \in \mathcal{R}$ and $u \notin \mathcal{S}$
- 11 Break ties by removing *background*, *observation*, *meta-reflection*, then *inference*
- 12 $\tau \leftarrow \tau + 1$
- 13 **return** $\mathcal{P}$, $L_{TSP^*} = |\mathcal{P}|$, τ

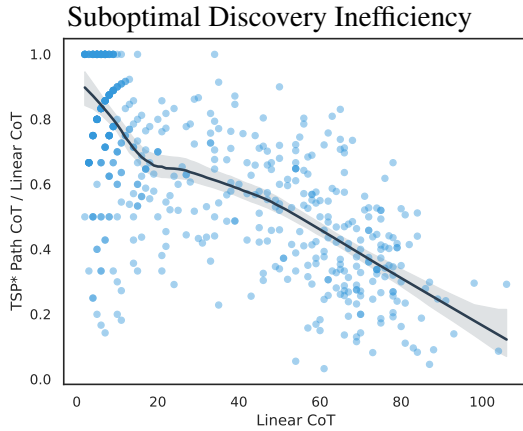


Figure 9: Scatter plot of loop-free CoT length (L_{loops} , x-axis) vs. inefficiency-based efficiency ($E = 1 - \mathcal{L}_{ineff.}$, y-axis), where $\mathcal{L}_{ineff.}$ captures only path inefficiency relative to L_{TSP^*} and excludes L_{loops} redundancy.

H Additional Figures and Tables

This appendix maps the additional diagnostics: Figure 10 covers step skipping, Table 8 covers relevance, Figure 11 covers attribution, Figure 12 covers Best-of- N complementarity, and Figure 13 shows the full example graph.

The node-level hallucination accuracies reported in §5.3 are aggregated over the 11 fixed splits described in Appx. A, using the same GINE model as the relevance task.

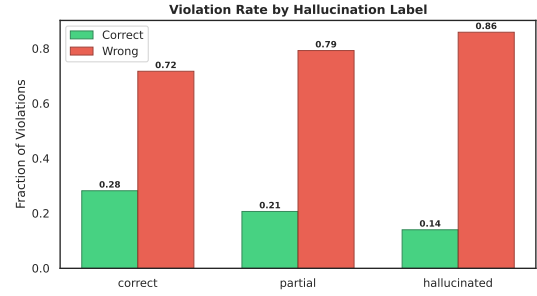


Figure 10: Comparison of step-skipping violation rate (see §5.5) across node categories (*correct*, *partial*, *hallucinated*) using LLM-as-Judge annotations (GPQA, DS-Qw14B). For each category, the bar is split between correct (green) and incorrect CoTs (red); similar results are obtained with Qwen-7B.

True \ Pred	Relevant	Redundant	Irrelevant
Relevant	65.2	34.8	0.0
Redundant	33.5	66.5	0.0
Irrelevant	21.1	78.9	0.0

Table 8: Confusion matrix (%) for node-level relevance prediction on GPQA (DS-Qw14B). Rows are ground-truth; columns are predicted labels.

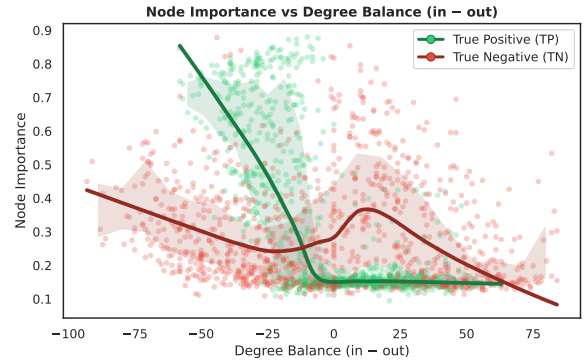


Figure 11: GNNExplainer node importance vs. (in-out)-degree values (GPQA, DS-Qw7B).

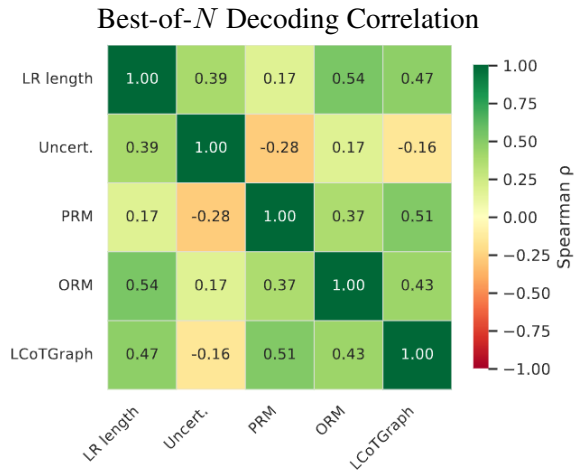


Figure 12: Spearman rank correlation matrix of Best-of- N selection scores on MMLU-Pro (DS-Qw7B, $N=5$).

I Prompts

The following prompts are used for CoT generation, LLM-as-Judge evaluation, and graph edge classification: Prompt 1 generates the CoT traces, Prompts 2a–c define the judge targets, and Prompt 3 defines graph edges.

Prompt 1 — CoT Generation

Solve the problem step by step.

Write the full reasoning process, with each step separated clearly from the next. Show all intermediate reasoning needed to reach the solution.

After completing the reasoning, write the final answer clearly at the end and do not generate additional text.

Problem:
{question}

Options:
A. {option_1}
B. {option_2}
...

Prompt 2a — LLM-as-Judge: Hallucination

TASK: Check each numbered step for factual accuracy, independently of other steps.

LABELS:

- correct: node fully accurate
- partial: node partially accurate, minor errors or incomplete
- hallucinated: factually wrong or unsupported

INPUT:
Problem: {problem}
Expected Solution: {solution}
Expected Final Answer: {answer}

Reasoning Steps: {cot}

INSTRUCTIONS: Evaluate all steps 0 to {cot_len-1} independently.

OUTPUT (ONE LINE PER NODE):
[i] <correct|partial|hallucinated>
| WHY: <3 words>

Prompt 2b — LLM-as-Judge: Relevance

TASK: Assess each step's contribution to the derivation of the final answer.

LABELS:

- relevant: contributes new information
- redundant: repeats prior information
- irrelevant: unrelated to solution

INPUT:
Problem: {problem}
Expected Solution: {solution}
Expected Final Answer: {answer}
Reasoning Steps: {cot}

INSTRUCTIONS: Analyze all steps 0 to {cot_len-1} in order.

OUTPUT (ONE LINE PER NODE):
[i] <relevant|redundant|irrelevant>
| WHY: <3 words>

Prompt 2c — LLM-as-Judge: Final Answer

TASK: Determine if the trace reaches the correct answer.

LABELS:

- final_answer_correct: <Correct, Incorrect>
- final_answer_in_cot: step numbers containing the final answer

INPUT:
Problem: {problem}
Expected Solution: {solution}
Expected Final Answer: {answer}
Reasoning Steps: {cot}

OUTPUT FORMAT (STRICT):
Final answer correct: <Correct, Incorrect>
Final answer in CoT: [comma-separated step numbers]

Prompt 3 - Edge Classification

You are given the original problem statement and two nodes from a reasoning process that attempts to solve it:

Problem Statement: {problem_text}

Node A ({src_idx}): {src_text}
Node B ({tgt_idx}): {tgt_text}

Your task: Evaluate the relationship from Node A to Node B. If there is a doubt about any relationship, choose "none" with strength 0.

Decide how Node A contributes to Node B by choosing exactly one of the following relationship types:

- "repetition": Node A and Node B express substantially the same or a very similar idea, result, conclusion, or reasoning step, even if phrased differently.
- "background info": Node A contains prior knowledge, context, assumptions, or general facts that are used in Node B.
- "inference": Node A provides an explicit premise, and Node B is just the direct logical consequence of it. Consider background info, observation, and meta-reflection instead if more appropriate.
- "observation": Node B is an observation, comment, or general step related to Node A, even indirectly.
- "meta-reflection": Node B involves task understanding, planning, decomposition, verification, self-checking, or reasoning about the problem-solving process rather than advancing the solution itself.
- "none": No meaningful relationship between Node A and Node B. This should be the default choice whenever the relationship is unclear, weak, speculative, or indirect.

Assign a strength score from 0 to 10:

- 0: no link - unrelated nodes
- 1-3: weak - indirect relation; Node A barely influences Node B
- 4-6: moderate - Node A contributes to Node B but is not required for it
- 7-8: strong - Node A is a primary driver or direct premise of Node B
- 9-10: essential - Node B directly and necessarily follows from Node A alone

Output exactly one valid JSON object, with no extra text:

```
{"class": <"meta-reflection" |  
  "repetition" | "inference" |  
  "background info" | "observation" |  
  "none">, "strength": <0-10>}
```

So, the JSON that classifies the pair is the following:

CoT2Graph: CoT → Reasoning Structure

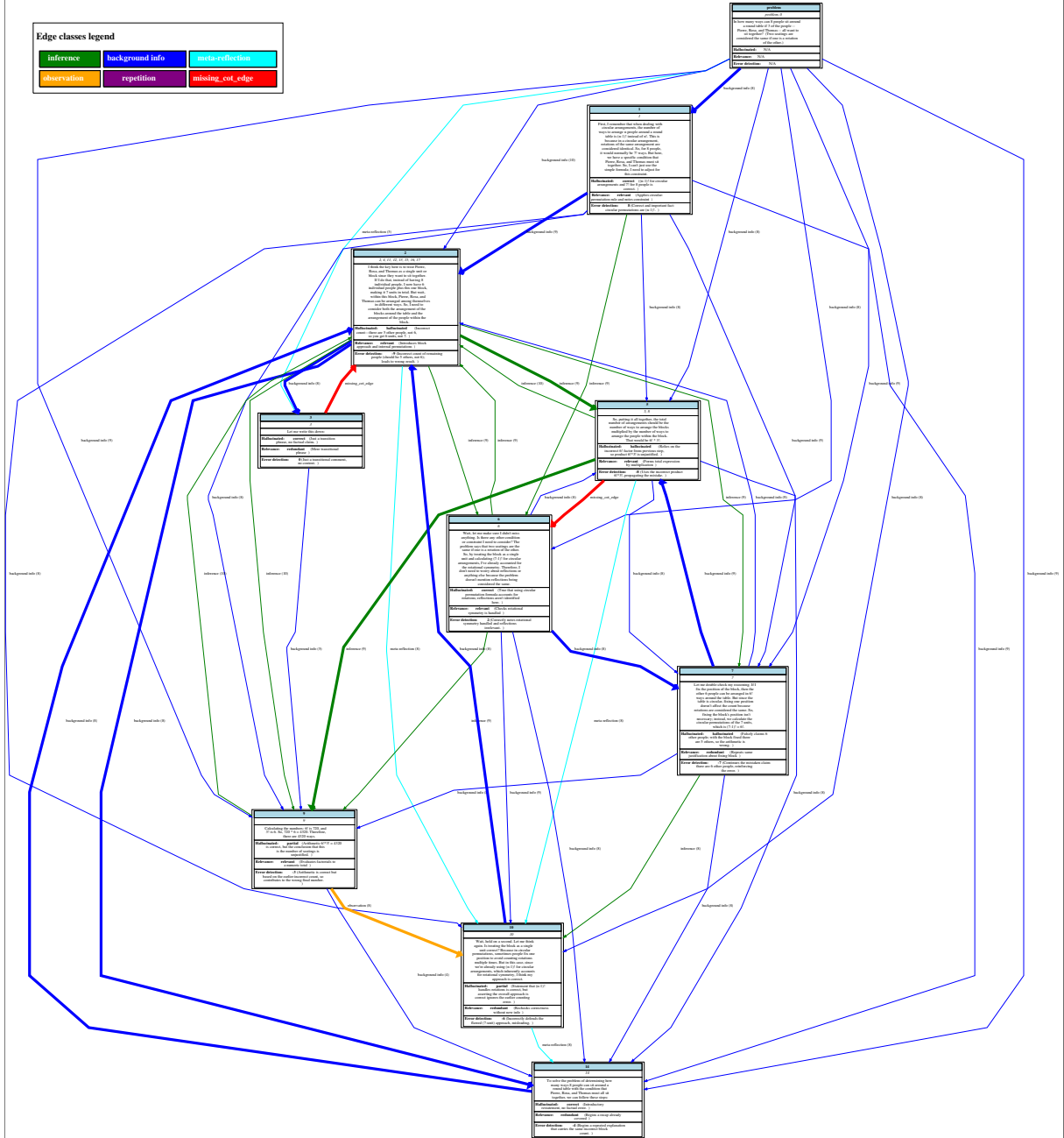


Figure 13: Full dependency graph for a representative GPQA (dataset entry ID: 32) trace (DS-Qw14B, correct prediction). Nodes correspond to atomic reasoning steps; edge colours encode type (*inference*: green, *background-info*: blue, *observation*: yellow, *meta-reflection*: light blue, *missing cot edge*: red). Repetition edges are not shown because their endpoints are merged into representative supernodes. Node 5, for example, represents the merged set $\{5, 8\}$ and marks a return to previously expressed content.