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

Anonymous ACL submission

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 taces. This raises a key question: what aspects of reasoning quality (i.e., correctness, efficiency, and failure patterns) is 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 subtasks merging. This enables structure-aware analysis of reasoning, capturing dependencies that are not accessible in sequence-based evaluations, and allows graph neural networks to learn how structural patterns of CoTs relate to their reasoning quality. CoT2Graph reveals key failure modes: many CoT steps are structurally inefficient, and errors such as dependency violations and overloaded synthesis steps arise from how reasoning is organized. We further show that graph-based signals improve prediction of reasoning outcomes, outperforming length-based baselines by up to +7.8% and 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.

1 Introduction

Models trained with reinforcement learning on reasoning tasks, such as DeepSeek-R1 (Guo et al., 2025) and GPT-5 Thinking (Singh et al., 2026), produce long Chain-of-Thought (LCoT) traces that involve backtracking over earlier conclusions, branching across concurrent sub-problems, and combining partial results from multiple reasoning threads (Wu et al., 2025). However, existing evaluation approaches treat these 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 information about *reasoning quality* such as the correctness of the final outcome, the efficiency of the reasoning process, and typical failure patterns are encoded in the structure of a Chain-of-Thought trace?

CoT2Graph: CoT → Reasoning Structure

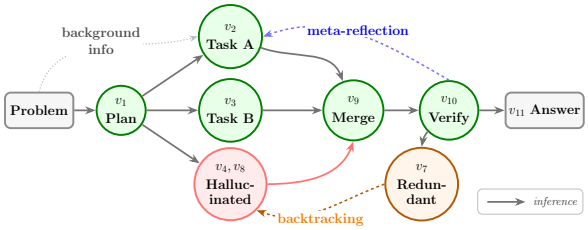

Figure 1: LCoT represented as an interpretable directed graph capturing the underlying reasoning process.

Step-level evaluation frameworks such as ROSCOE (Golovneva et al., 2023) and ReCEval (Prasad et al., 2023) effectively detect local errors (e.g., unsupported inferences or factual mistakes), but are insufficient for capturing global reasoning structure, as they evaluate steps in isolation. Existing approaches do not test whether such structural patterns can be used to predict or explain reasoning quality. Existing approaches (Dalvi et al., 2021; Jiang et al., 2025b) represent CoT traces either as simplified structural forms such as linear sequences or trees. However, these representations are insufficient for capturing the dependency structure of complex reasoning processes. In particular, they cannot model multi-step dependencies across distant parts of the trace, the aggregation of multiple reasoning threads, or the revisitation and reuse of intermediate results. As a result, they fail to capture structural patterns that are critical for understanding reasoning quality, including how errors propagate, how redundancy emerges, and how reasoning steps depend on one another.

To address this question, we introduce

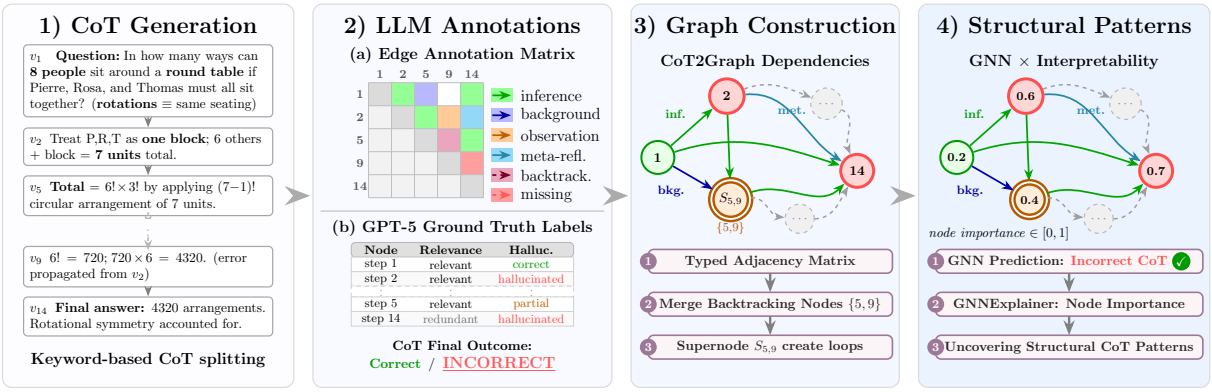

Figure 2: CoT2Graph pipeline for graph construction and reasoning trace structural analysis ($GPQA_{ID\ 32}$, Figure 12).

CoT2Graph, a framework that represents CoT traces as directed dependency graphs and models them using trained graph neural networks (GNNs). In this representation, nodes correspond to reasoning steps and edges encode their relationships such as inference, backtracking, observation, and meta-reflection. GNNs provide a natural mechanism for capturing dependency structure through message passing over nodes and typed edges, allowing the model to aggregate information across multiple reasoning steps, incorporate heterogeneous transition types, and model multi-hop dependencies. As a result, the framework can relate both local and global structural patterns to CoT correctness.

Finally, GNNs also provide a natural degree of interpretability, allowing us to identify which nodes and transition types contribute most to prediction outcomes. Using GNNExplainer, a popular approach for explaining GNN predictions, we empirically show that reasoning structure encodes rich signals about reasoning quality. We provide new insights into reasoning behavior, showing that a large fraction of CoT steps are structurally inefficient, and identifying failure modes such as intermediate step skipping and overloaded synthesis nodes. Finally, we extend existing semantic evaluation frameworks with structural errors that link empirical graph patterns to LLMs failures.

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 hierarchical 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, 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), and 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., 2025; Xue et al., 2024). Our work adds two structural error categories to the semantic taxonomy, shows that step redundancy is predictable from graph structure, and provides an efficiency decomposition that separates productive exploration-verification pattern from inefficiency.

3 CoT2Graph Construction Pipeline

CoT2Graph converts Chain-of-Thought (CoT) traces into structured dependency graphs and models them using graph neural networks (GNNs) to analyse reasoning quality. The pipeline consists of four stages: *(i)* generating CoT traces, *(ii)* segmenting them into atomic reasoning steps, *(iii)* annotating dependencies between them, and *(iv)* constructing a graph representation (Figure 2). This representation captures non-linear reasoning patterns such as branching, merging, and revisitation, while the integrated GNN models enable prediction and analysis of reasoning quality, including correctness, efficiency, and failure modes.

3.1 Graph Representation

We represent each CoT trace as a directed graph $G = (V, E, \phi, \psi)$, where nodes $V = \{v_1, \ldots, v_n\}$ represent atomic reasoning steps and edges $E \subseteq V \times V$ capture dependencies between them. Edges are assigned types that capture distinct reasoning transitions, including inference, background knowledge, observation, meta-reflection, and backtracking. This allows the graph to encode not only which steps are connected, but how they interact within the reasoning process. Each node is associated with structural features (e.g., position and degree statistics), and each edge is annotated with a type and strength score. Together, this representation transforms CoT into a graph that captures multi-step dependencies and non-linear reasoning patterns.

3.2 Generation of CoT Traces

We generate Long CoT 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 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 length of 16K tokens. This configuration produces stable yet sufficiently diverse reasoning traces for downstream analysis.[1]

3.3 Node Segmentation

Each CoT trace is segmented into atomic reasoning steps $V = \{v_1, \ldots, v_n\}$ using a rule-based parser that identifies structural transition markers (e.g., "Therefore", "However", "Finally", etc) via regex matching. Fragments shorter that 100 characters are merged to produce sequences of meaningful reasoning units. The resulting sequence preserves the original generation order while producing coherent reasoning units for downstream analysis.

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* (logical deduction), *background* (contextual knowledge), *observation* (evidence extraction), *meta-reflection* (reasoning about prior steps), *backtracking* (semantic equivalence), and *none* (no meaningful dependency). 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

The final reasoning graph is constructed by combining nodes and annotated edges. To consider repeated reasoning as well, nodes connected by backtracking edges are merged using Union-Find algorithm, grouping semantically equivalent steps into equivalence classes. When merging nodes, multiple incoming edges between equivalence classes may induce conflicting edge types. Concretely, we retain the edge with the highest strength score and if multiple edges have equal strength, we resolve ties using a fixed priority ordering over edge types (background $\prec$ observation $\prec$ meta-reflection $\prec$ inference), favoring edges that represent stronger logical dependencies. Each equivalence class is represented by a canonical node, and all incident edges are redirected accordingly. This collapses repeated reasoning steps into a single representation while preserving the overall dependency structure. This compression introduces cyclic structures that reflect revisitation and verification behavior, capturing non-linear reasoning patterns in the graph.

[1] Further details are provided in (§3.7).

Step-Level Annotation via LLM-as-Judge Finally, we annotate each reasoning step using an LLM-as-judge 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. We construct step-level supervision labels using GPT-5 Mini (Singh et al., 2026) as an LLM judge, and additionally validate a subset through human annotation to ensure label quality. These annotations jointly serve as (i) supervision signals for training the GNN models (§4.1), (ii) pruning labels for the efficiency analysis in §4.4, and (iii) ground truth for Best-of-N selection in §4.5. LLM-as-Judge prompts and details are provided in Appx. A, D.

3.6 GNN Modeling and Structural Analysis

Given the constructed graphs, we train graph neural networks (GNNs) (Xu et al., 2019) on the constructed graphs to learn how structural patterns in reasoning traces relate to reasoning quality. Unlike sequence-based models, GNNs explicitly operate over the graph structure, supporting the aggregation of information across multiple reasoning steps and the modeling of multi-hop dependencies. GNNs incorporate both node features (e.g., position and degree statistics) and typed edges, allowing the model to condition on different reasoning transitions (e.g., inference, meta-reflection). This allows us to model and analyse how local structures (e.g., high in-degree nodes) and global graph patterns jointly influence reasoning outcomes. We consider both graph- and node-level tasks, and filter the dataset to maintain a balanced GNN training set with comparable numbers of correct and incorrect CoT graphs. For graph-level prediction, a GATv2 model (Brody et al., 2022) predicts final-answer correctness from the full reasoning graph. For node-level prediction, a GINE model (Hu et al., 2020) classifies individual reasoning steps by relevance and hallucination, enabling fine-grained analysis of reasoning behavior. These learned models enable both prediction of reasoning outcomes and extraction of structural signals that inform our downstream analysis.[2]

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.

3.7 Experimental Setup

Evolution Datasets and LLMs. Primary experiments use GPQA (Rein et al., 2023), 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 Team, 2025), trained with a direct reinforcement learning on verifiable rewards. In Appx. B, we provide additional LLM post-training details relevant for cross-model generalization discussion.

4 Results and Discussion

Graph-structured representations offer a lens to understand how reasoning processes develop in long CoT. Studying how current LLMs reason is crucial to identify where and why they fail, particularly in multi-step reasoning tasks where hallucinations and compounding errors remain a central challenge in AI safety. At the same time, efficiency is also important, as unnecessarily long traces translate into higher computational cost. Unlike surface-level metrics such as token length or final-answer accuracy, CoT2Graph uncovers the dependencies between reasoning steps, enabling fine-grained analysis of correctness (§4.1), structural analysis (§4.2), reasoning efficiency (§4.4), and error patterns in reasoning traces (§5). Finally, we discuss its applications in Best-of-N decoding (§4.5) and RL-reward shaping for LLM post-training.

4.1 CoT2Graph Predictive Learnable Signals

We first evaluate the capacity to predict whether a given CoT trace leads to a correct final an-

[2]Details on training procedures and the architectural choices are provided in Appx. A.

swer, framed as a binary graph-level classification task. Across all evaluation settings, CoT2Graph consistently outperforms both baselines, length-based outcome prediction and LCot2Tree, achieving gains of up to +7.8% over the length-based regression baseline (see Table 1). The advantage over LCoT2Tree is clearly visible across all model families, demonstrating that the richer representational capacity of graphs (which can encode cycles, synthesis steps, cross-step dependencies, and non-hierarchical chains) translates into measurable predictive gains. Notably, we observe the largest margin for DS-Qw14B (+7.8%), suggesting that larger models produce more complex reasoning structures, which contain predicting signals.

4.2 Step-by-Step CoT Structural Analysis

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* step archives 65.2% class accuracy on predicting relevant nodes and 66.5% on redundant nodes (see Table 5), indicating that graph structure encode a significant signal for detecting redundancy in CoT. Irrelevant nodes are grouped with redundant (less than 1% of dataset).

Hallucination is an Intra-Step Phenomenon. In contrast, graph structure is not sufficient to localize hallucinated[3] steps in the trace. The GNN can capture global signals of failure at the graph level, but achieves only 7.0% class accuracy on partially hallucinated nodes and 7.9% on incorrect ones.

This results suggests a separation between outcome-level and step-level hallucination detection: structural signals are informative for overall correctness, but intra-step hallucinations are mainly driven by nodes semantic content. We further analyse these patterns in §4.4 and §4.5.

4.3 Structural Interpretability

Early Planning Steps predict Correctness. The GNNExplainer (§3.6) reveals a strong asymmetry between correct and incorrect reasoning traces. In case of true positive predictions, node importance concentrates on the first 20–25% of the steps of a CoT (Figure 3). Further, manual analysis reveals that these early nodes of a reasoning trace primarily contain problem formalization and planning steps, including decomposing the task into sub-problems and deciding the overall reasoning strategy.

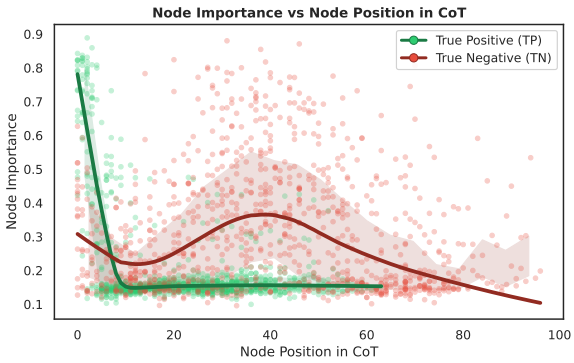

Figure 3: GNNExplainer node importance across CoT nodes positions (GPQA, DS-Qw7B). Early nodes set the reasoning direction, while intermediate nodes remain critical as they often introduce errors that propagate to the final answer.

Similarly, true negative predictions also show a concrete positional pattern. Node importance shifts toward the middle of the CoTs, suggesting that incorrect traces are often associated with flawed structure at the level of intermediate reasoning steps. These mid-chain errors propagate through subsequent steps often not captured by verification loops, leading to an incorrect final answers.

Planning and Synthesis Steps Are Crucial. We next examine how node connectivity (in-degree and out-degree)[4] relates to final outcome correctness (see Figure 10). 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. We further confirm this pattern through manual inspection of CoT traces. Overall, correct reasoning shows initial planning and more distributed aggregation, while incorrect CoTs are characterized by dense synthesis steps. This suggests that these structural differences could be leveraged during post-training to encourage more robust reasoning structures.

[3] In Appx. D, we defined *hallucination* as an errors that arise within a step itself (e.g., incorrect computation, incorrect law, etc), rather than flawed dependencies on previous nodes.

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

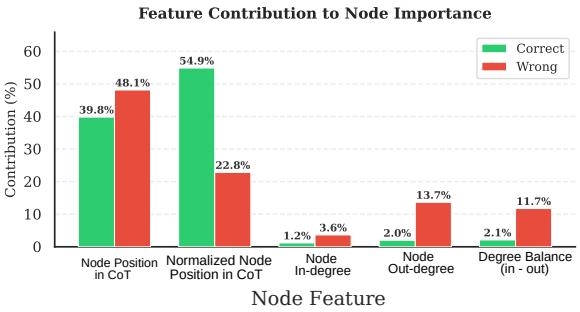

Figure 4: Linear regression of node features importance scores, comparing correct and incorrect predicted CoTs.

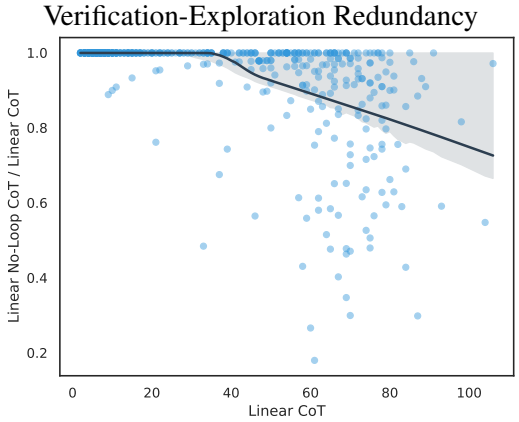

Figure 5: Scatter plot of original CoT length (L_{CoT}, x-axis) vs. efficiency ($E = 1 - L_{\text{loss verif.}}$, y-axis), where $L_{\text{loss,verif}}$ includes only good redundancy (verification/exploration) and excludes path inefficiency. We observe that verification loops starts appearing in CoTs long as least 35–40 steps, and then efficiency decreases approximately linearly as CoT length increases.

Feature Attribution via Regression Analysis. Regression analysis over GNNExplainer scores confirms the structural trends identified above. In correct CoTs, normalized node position is the dominant signal (54.9% vs. 22.8% in incorrect traces), indicating that GNN importance is more strongly driven by early-stage planning structure in successful reasoning. In contrast, incorrect CoTs rely more on absolute position, consistent with the fact that erroneous traces are typically longer (Wu et al., 2025). Node connectivity becomes more relevant in incorrect CoTs, suggesting that failures are more associated with intermediate subtasks aggregation.

4.4 Reasoning Efficiency Analysis

Reasoning efficiency is important for both LLMs performance and practical constraints such as computation cost, as overly long traces can degrade outcomes. However, long CoTs are not uniformly efficient: while exploration and verification can improve performance (Wang et al., 2023; Lightman et al., 2023), reasoning LLMs often generate unnecessarily long or even endless CoTs. Disentangling these components is important for understanding model behavior and for constructing high-quality post-training data. Since human CoT evaluation does not scale, we can rely on the graph structure of CoTs to automatically annotate exploration-verification and inefficient steps.

Structural Efficiency Decomposition. We quantify efficiency by decomposing reasoning traces into productive and redundant components.

Let L_{CoT} denote the total number of steps in the linear CoT, L_{loops} the length after collapsing backtracking loops, and L_{TSP^*} the length of an approximate minimal dependency-consistent path through the CoT graph – inspired by the travel salesman problem (TSP). We define CoT efficiency $\mathcal{E}$ and efficiency loss $\mathcal{L}$, where $\mathcal{E} = 1 - \mathcal{L}$, as:

$$\mathcal{E} = 1 - \left[\underbrace{\frac{L_{\text{CoT}} - L_{\text{loops}}}{L_{\text{CoT}}}}_{\mathcal{L}_{\text{verification}}} + \underbrace{\frac{L_{\text{loops}} - L_{\text{TSP}^*}}{L_{\text{CoT}}}}_{\mathcal{L}_{\text{path ineff.}}} \right] \quad (1)$$

Graph-Based Optimal TSP* Path Construction. We approximate the optimal L_{TSP^*} exploration path by constructing a dependency-consistent traversal of the CoT graph detailed in Algorithm 1 (see Appx. C). Starting from the problem node, we iteratively visit nodes whose prerequisites (incoming edges) are already discovered, enforcing a topological ordering over the graph. This ensures that each step is executed only after all required information is available. When a circular dependency prevent a valid traversal, we iteratively remove the weakest dependency edges until the graph becomes acyclic enough to allow progression. This relaxation is necessary because CoT graphs may contain cyclic structures induced by backtracking and verification behavior, which are not errors but part of exploration dynamics that we will use later to analyse the intermediate step skipping phenomenon.

Empirical Findings. Table 2 reports the decomposition of reasoning redundancy across models and CoT length quantiles. Path efficiency loss $\mathcal{L}_{\text{ineff.}}$ is the dominant component in all models, averaging 49.0% of steps compared to 6.6% for good redundancy $\mathcal{L}_{\text{verif.}}$, indicating that many reasoning steps are unnecessary. Good redundancy

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.

(verification/exploration) mainly arises in longer reasoning traces. As shown in Figure 5, it is negligible in short CoTs and appears only after 35–45 steps, after which it increases approximately linearly with CoT length. In contrast, path inefficiency is present even in short CoTs and grows approximately linearly with length (Figure 8). Longer traces accumulate increasingly misordered steps, leading to higher inefficiency across all quantiles. Larger model shows lower inefficiency while maintaining similar levels of verification redundancy. Good redundancy and path inefficiency are positively correlated: traces with more verification also tend to include more structural inefficiency. This reflects a trade-off at generation time: verification increases coverage but also introduces suboptimal step ordering.

Skipping Intermediate Steps and Hallucination. Complex reasoning often requires decomposing out-of-distribution problems into intermediate subtasks 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. Node skipping is defined as the relaxation events in the L_{TSP^*} path construction pipeline (§4.4, Algorithm 1). Specifically, when dependency constraints prevent a valid traversal, edges are removed to resolve blocking cycles and allow the algorithm to proceed. As shown in Figure 9, step-skipping increases with node-level error severity. Compared to correct nodes, partially correct nodes show a +6.96% higher skipping rate, while hallucinated nodes increase by a further +7.03%, consistent across models. This indicates that a frequent bypassing of intermediate steps contribute to intra-node hallucination and final outcome errors.

4.5 CoT2Graph Best-of-N Decoding

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

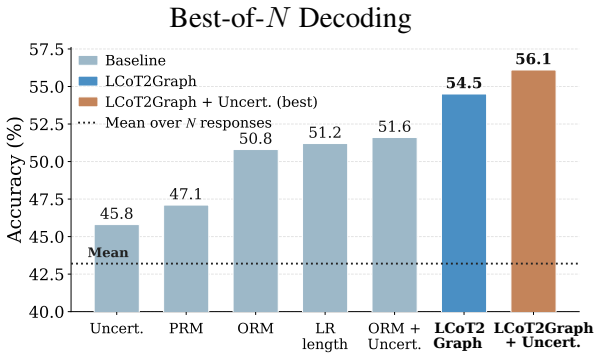

Figure 6: Best-of-N ($N = 5$) selection accuracy on MMLU-Pro with DS-Qw7B. CoT2Graph-Best achieves the highest performance among single verifiers (54.5%) with a baseline of 43.2%, outperforming all previous methods by at least 3.3%. Combining CoT2Graph with entropy selection further improves accuracy to 56.1%.

CoT Verifiers Methods. We compare five selection strategies. Entropy-based method selects the CoT with lowest mean per-token entropy (Fu et al., 2025). PRM-Best (Mistral-7B-PRM) aggregates step-level rewards (Lightman et al., 2023; Wang et al., 2024a), while ORM-Best (Skywork-Reward-Llama-8B) uses a final outcome score (Liu et al., 2026). A logistic regression ranks CoTs by length as a baseline reference. CoT2Graph-Best ranks CoTs using GNN-predicted correctness. For fair comparison, the verifier models used in PRM, ORM, and the CoT2Graph edge annotation pipeline are all in the 7–8B parameter range.

We evaluate on 600 MMLU-Pro questions using DS-Qw7B and N=5 sample. The mean baseline achieves 43.2%, while the oracle upper bound s 68.3%. CoT2Graph-Best reaches 54.5%, outperforming all previous methods including ORM-Best (50.8%) and PRM-Best (47.1%) just from the graph structure of the reasoning process (see Figure 6).

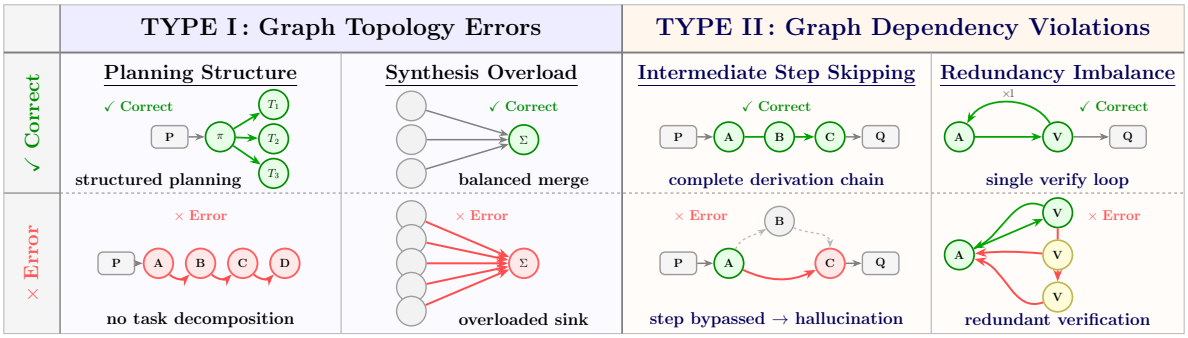

Figure 7: Structural reasoning patterns identified by the **CoT2Graph** proposed taxonomy, including planning structures, synthesis node overload, intermediate step skipping, and verification redundancy.

Orthogonal Signals and Ensemble Strategy. The correlation analysis in Figure 11 shows two regimes. CoT2Graph is weakly correlated with entropy-based/uncertainty selection ($\rho = -0.16$), consistent with our analysis in §4.2, where node-level hallucination was found to be an intra-step phenomenon not captured by structural signals. In contrast, it shows moderate correlation ($\rho > 0.4$) with PRM, ORM, and length-based ranking, indicating that our method already leverage the predictive signal used by other verifiers. This suggests that CoT2Graph and entropy method capture complementary failure modes. Accordingly, the ensemble improves performance to 56.1% (+1.6%).

5 A Structural Error Taxonomy for Chain-of-Thought Reasoning

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. However, they do not capture failures arising from the underling structure of the CoT (see Figure 7). We extend this view with two structural error categories derived from our graph analysis. These patterns capture failure modes linked to reasoning organization rather than individual step correctness.

Type I: Graph Topology Errors. Errors arising from the flawed structure of the reasoning trace. **Planning structure.** Correct CoTs often contain early planning nodes with moderate out-degree, corresponding to task decomposition and subtasks 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 in once. These nodes are strongly associated with reasoning failures and structurally fragile synthesis steps.

Type II: Graph Dependency Violations. These errors arise when reasoning steps violate the dependency structure of the graph. **Intermediate step skipping.** Some CoTs bypass required intermediate derivations, producing conclusions before prerequisite subgoals are established. These violations are associated with hallucinated nodes. **Redundancy imbalance.** Verification and exploration loops can improve reasoning when used moderately, but excessive branching, repeated verification, or endless loops introduce structural inefficiency and increase failure probability.

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.

6 Conclusion

We introduced CoT2Graph, a graph-based representation of Long Chain-of-Thought reasoning that makes dependencies between steps explicit. Structural signals provide a more informative view of reasoning than sequence-based approaches, improving both outcome prediction and interpretability for failure analysis. Our results show that reasoning errors often arise from structural issues, such as misordered dependencies or overloaded synthesis steps, and that a large fraction of reasoning traces is inefficient. Overall, this work motivates a shift from sequence-level evaluation to structure-aware analysis of reasoning processes and models that reason both correctly and efficiently.

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 evaluator (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.

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. Our framework is intended to support interpretability and the analysis of Chain-of-Though reasoning patterns in large language models, with the goal of better understanding why failures and hallucinations occur.

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.

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 Hoefler. 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? *Preprint*, arXiv:2105.14491.

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. *Preprint*, arXiv:2503.19470.

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.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. 2025. Deep think with confidence. *Preprint*, arXiv:2508.15260.

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.

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, Roee 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. *Preprint*, arXiv:2502.09621.

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, 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.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai 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.

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, Jiaxi Yang, Jingren Zhou, and 25 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. 2023. GPQA: A graduate-level google-proof Q&A benchmark. *Preprint*, arXiv:2311.12022.

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 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.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Preprint*, arXiv:2406.01574.

Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025. When more is less: Understanding chain-of-thought length in llms. *Preprint*, arXiv:2502.07266.

Xiaomi LLM Team. 2025. MiMo: Unlocking the reasoning potential of language model – from pretraining to posttraining. *Preprint*, arXiv:2505.07608.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? *Preprint*, arXiv:1810.00826.

Xin Xu, Shizhe Diao, Can Yang, and Yang Wang. 2025. Can we verify step by step for incorrect answer detection? *Preprint*, arXiv:2402.10528.

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.

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

1 Introduction **1**

2 Related Work **2**

3 CoT2Graph Construction Pipeline **3**
3.1 Graph Representation 3
3.2 Generation of CoT Traces 3
3.3 Node Segmentation 3
3.4 Edge Annotation 3
3.5 Graph Construction 3
3.6 GNN Modeling and Structural Analysis 4
3.7 Experimental Setup 4

4 Results and Discussion **4**
4.1 CoT2Graph Predictive Learnable Signals 4
4.2 Step-by-Step CoT Structural Analysis 5
4.3 Structural Interpretability 5
4.4 Reasoning Efficiency Analysis . . 6
4.5 CoT2Graph Best-of-N Decoding . 7

5 A Structural Error Taxonomy for Chain-of-Thought Reasoning **8**

6 Conclusion **8**

A CoT2Graph Implementation Details **12**
A.1 LLM Generation Hyperparameters 12
A.2 Node Segmentation 12
A.3 Node Feature Vector 12
A.4 Edge Classification 12
A.5 Graph Compression via Union-Find 12
A.6 GNN Architecture and Training . 12

B Cross-Model Generalization **13**
B.1 Model Characterization 13
B.2 Cross-Model Transfer Results . . 13

C Optimal TSP* Path Details **14**

D Prompts **16**

A CoT2Graph Implementation Details

This appendix provides implementation details of the CoT2Graph framework, covering the full process from LLMs hyperparameters to 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 similarly as in previous works (Jiang et al., 2025b). Segments shorter than 100 characters are merged with the following step to ensure semantic completeness. The resulting sequence $v_1, v_2, \ldots, v_n$ preserves linear generation order.

A.3 Node Feature Vector

Each node v_i is represented by a six-dimensional feature vector:

$$\phi(v_i) = \big[i,\ i/|\mathcal{V}|,\ d_i^{\text{in}},\ d_i^{\text{out}},\ d_i^{\text{in}} - d_i^{\text{out}},\ d_i^{\text{in}}/(d_i^{\text{out}} + \varepsilon)\big],$$

encoding absolute position, normalized position, in-degree, out-degree, degree balance, and degree 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 halves prompt cost and encodes a causal prior consistent with sequential generation of the LLM. In Appx. D we included 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), *backtracking* (v_i and v_j cover essentially the same content), and *none* (no meaningful dependency).

A.5 Graph Compression via Union-Find

Nodes connected by *backtracking* edges are merged using a Union-Find data structure. Each equivalence class is represented by a canonical node, and all incident edges are redirected via a node-to-representative mapping. This collapses CoT steps that restate the same reasoning content into single canonical nodes, introducing loop structures in the graph while preserving unique conceptual content.

A.6 GNN Architecture and Training

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($64 \to 64$)–ReLU–Dropout(0.2)–Linear($64 \to 2$)). 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. 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. All GNN models use a 70/15/15 train/validation/test split. Results are averaged over five independent random splits to mitigate sensitivity to data partitioning.

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 3: 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 (↓) / Test (→)	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 *(ours)*	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 4: 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.

B Cross-Model Generalization

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's robustness, this analysis reveals how post-training strategy and model scale shape underling reasoning structure.

B.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 3 summarizes their principal characteristics.

DS-Qw7B & 14B. Both models are initialized from the Qwen2.5 (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 → RL → SFT → 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 Team, 2025) is quite distant from DeepSeek. Its pre-training curriculum spans across 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.

B.2 Cross-Model Transfer Results

Table 4 reports F1 scores for graph-level correctness classification. We report macro F1 as comparison metric rather than accuracy throughout 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 → 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. Moreovere, LCoT2Tree requires a frontier model (e.g., DeepSeek-v3) to produce the intermediate sketch used for tree construction, introducing a costly and model-dependent pre-processing 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 (which exhibit greater topological diversity, more frequent exploration loops, and more varied synthesis node configurations) learns a generalizable structural vocabulary of both correct and incorrect reasoning. Conversely, a GNN trained on the more homogeneous distillation traces of DS-Qw7B transfers less effectively to the broader structural pstterns of MiMo-7B-RL.

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.

C Optimal TSP* Path Details

This section provides full details of the graph-based optimal path construction procedure used to estimate $L_{\text{TSP}*}$, including the traversal algorithm and dependency relaxation mechanism (Algorithm 1).

We also include additional plots (Figure 8) illustrating how CoT efficiency $E = 1 - L_{\text{TSP}*}$ degrades linearly with increasing CoT length when verification loops L_{loops} are not considered. Figure 9 compare step-skipping violations across node categories correctness under LLM-as-Judge labels.

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

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

Remove all nodes labelled *redundant* or *irrelevant*
$\mathcal{S} \leftarrow \{v_0\}$ ▷ visited; v_0 = problem node
$\mathcal{R} \leftarrow \mathcal{V} \setminus \{v_0\}$; $\mathcal{P} \leftarrow [v_0]$; $\tau \leftarrow 0$
while $\mathcal{R} \neq \emptyset$ **do**
  $\mathcal{F} \leftarrow \{v \in \mathcal{R} \mid \forall (u,v) \in \mathcal{E} : u \in \mathcal{S}\}$ ▷ nodes with all prerequisites satisfied
  if $\mathcal{F} \neq \emptyset$ **then**
    $v^* \leftarrow \arg\min_{v \in \mathcal{F}} \mathrm{id}(v)$
    Append v^* to $\mathcal{P}$; $\mathcal{S} \leftarrow \mathcal{S} \cup \{v^*\}$; $\mathcal{R} \leftarrow \mathcal{R} \setminus \{v^*\}$
  else ▷ deadlock: relax one prerequisite edge
    Remove from $\mathcal{E}$ the lowest-priority blocking edge (u,v), $v \in \mathcal{R}$, per order: *background* $\succ$ *observation* $\succ$ *meta-reflection* $\succ$ *inference*
    $\tau \leftarrow \tau + 1$
  end if
end while
return $\mathcal{P}$, $L_{\text{TSP}*} = |\mathcal{P}|$, τ

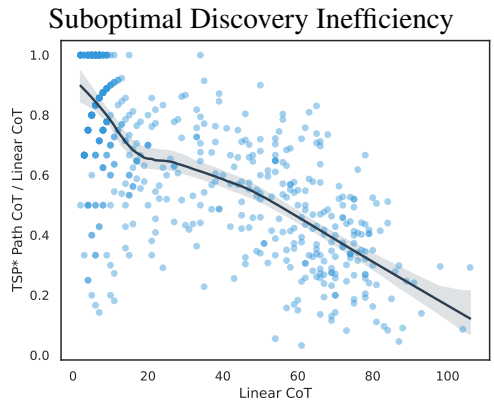

Figure 8: Scatter plot of loop-free CoT length (L_{loops}, x-axis) vs. inefficiency-based efficiency ($E = 1 - \mathcal{L}_{\text{ineff}}$, y-axis), where $\mathcal{L}_{\text{ineff}}$ captures only path inefficiency relative to $L_{\text{TSP}*}$ and excludes L_{loops} redundancy.

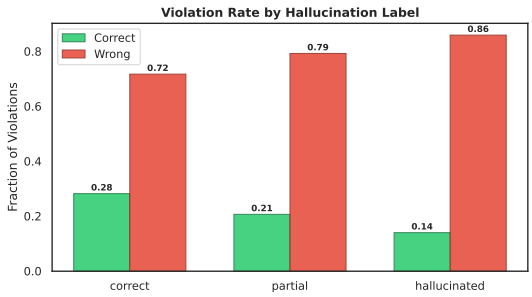

Figure 9: Comparison of step-skipping violation rate (see §4.4) across node categories (*correct, partial, hallucinated*) using LLM-as-Judge annotations (GPQA, DS-Qw14B). For each category, the bar is split between correct (green) and incorrect CoTs (red), similar results are obtained with Qwen-7B.

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

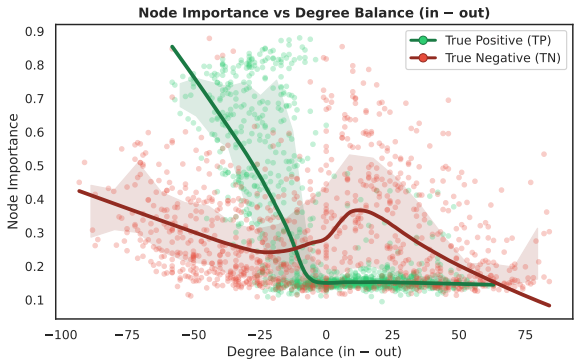

Figure 10: GNNExplainer node importance vs (in-out)-degree values (GPQA, DS-Qw7B).

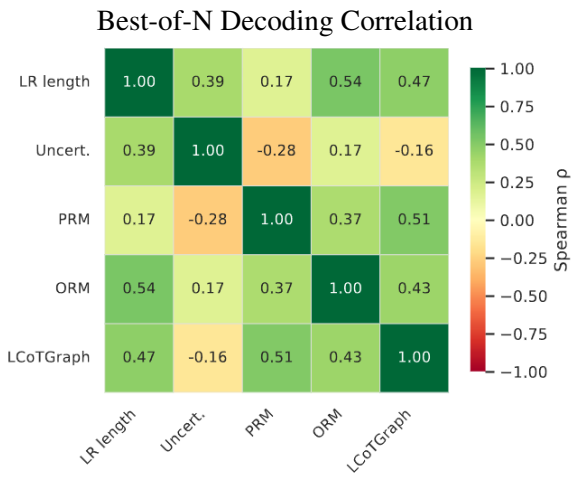

Figure 11: Spearman rank correlation matrix of Best-of-N selection scores on MMLU-Pro (DS-Qw7B, N=5).

D Prompts

The following are the prompts are used for CoT generation, LLM-as-Judge for CoT evaluation and edge classification for graph construction.

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 two reasoning nodes:
Node A (#{src_idx}): {src_text}
Node B (#{tgt_idx}): {tgt_text}

Classify how Node A contributes to
Node B. Choose exactly one type:
- "meta-reflection": planning or
  verification
- "backtracking": near-identical
  content
- "inference": logical or mathematical
  deduction
- "background info": prior knowledge
  relevant to Node B
- "observation": Node B supported
  by Node A
- "none": no meaningful relationship

Also assign a strength: 0 (weak)
to 10 (strong).

Output one JSON, no extra text:
{"class": <type>, "strength": <0-10>}
```

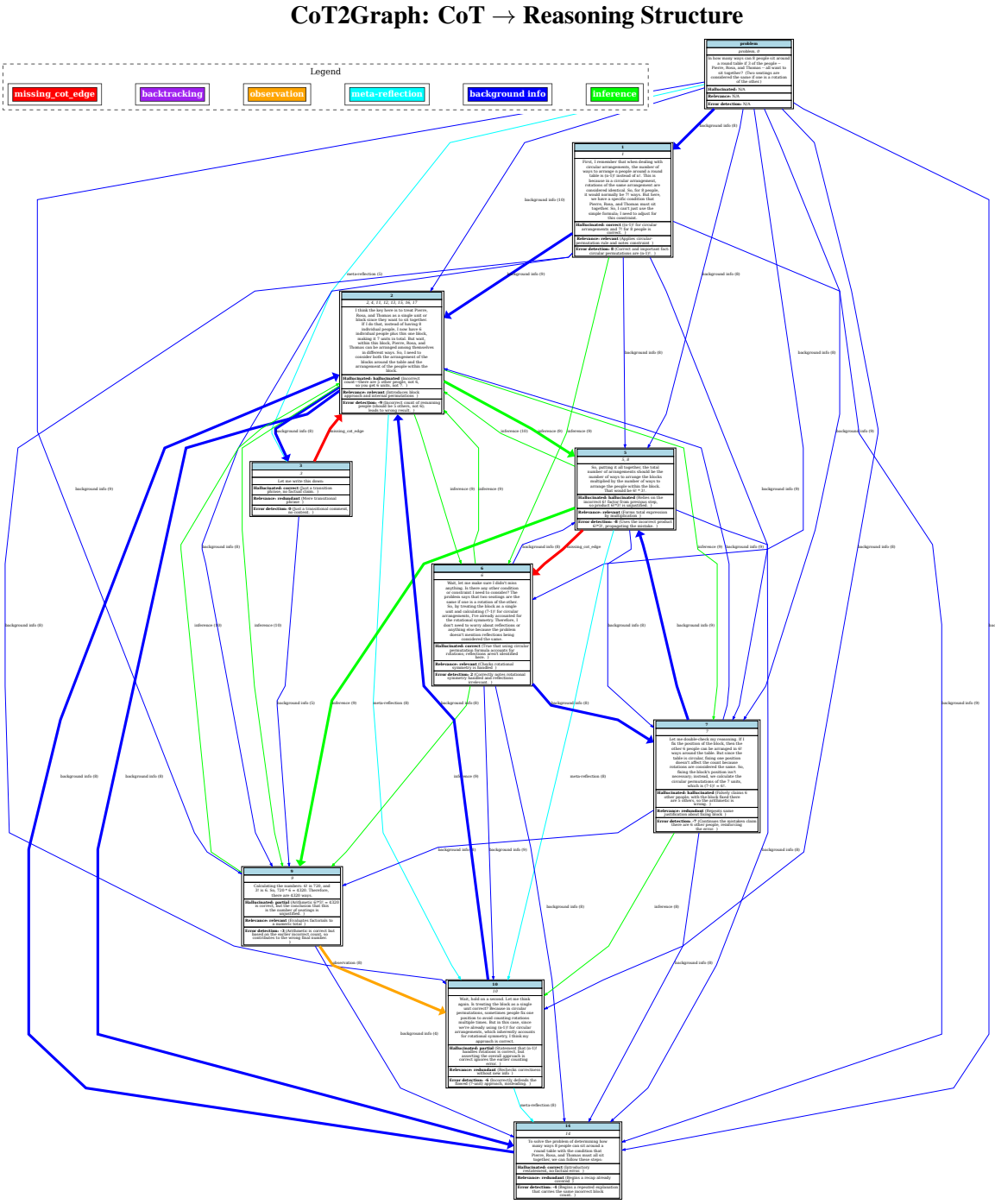

Figure 12: Full dependency graph for a representative GPQA (dataset entry ID: 32) trace (DS-Qw14B, correct prediction). Nodes correspond to atomic reasoning steps; edge colours encode type (*inference*: green, *background-info*: blue, *observation*: yellow, *meta-reflection*: light blue, *missing-edge*: red). Backtracking edges are not shown, as nodes connected through *backtracking* relations are merged into representative supernodes during graph construction. For example, node 5 represents the merged set $\{5, 7\}$, indicating that the reasoning trace returned to previously explored content.