

Structural Chain-of-Thought: Reasoning with Executable Computational Graphs

Petr Senatorov¹, Karim Vafin¹, Elena Tutubalina², Oleg Somov^{1,2}
¹MIPT, ²AIRI,

Correspondence: senatorov.pp@phystech.edu, somov@airi.net

Abstract

Chain-of-thought prompting enables reasoning in large language models but produces explanations that are difficult to verify, expensive to generate, and not always faithful. We introduce a Structural Chain-of-Thought framework where models generate executable computational graphs instead of free-form reasoning. The final answer is obtained by deterministically executing the graph. The model is trained using graph-based reinforcement learning that rewards structural validity, answer correctness, and node overlap with reference graphs.

Experiments across five benchmarks and four model families show that our method consistently improves accuracy over supervised graph training and can reduce generation length by up to 5-6× compared with standard GRPO and chain-of-thought supervised training. Our faithfulness experiments reveal that models trained with our approach tend to rely on deriving their answers from the generated graphs rather than producing post-hoc rationalizations as usually in chain-of-thought reasoning.

1 Introduction

Large language models have demonstrated strong reasoning abilities across a wide range of tasks by generating intermediate reasoning steps before producing a final answer (Wei et al., 2022; Wang et al., 2023a; Kojima et al., 2022). However, these reasoning traces can create the impression that the model arrived at its answer through the displayed reasoning process, but recent works have raised concerns about the faithfulness of such traces, showing that generated explanations do not always reflect the model’s true internal decision process (Lanham et al., 2023; Turpin et al., 2023; Tutek et al., 2025; Han et al., 2026).

A likely contributing factor is the training objective itself: modern reasoning models are typically trained using rewards based solely on answer correctness and output format, without imposing any

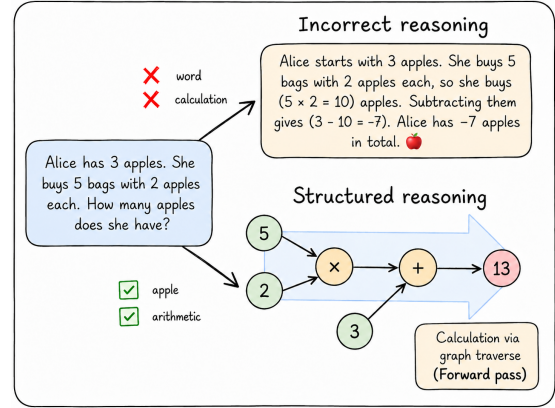


Figure 1: Comparison between free-form reasoning and computational graph representations for solving a natural-language arithmetic problem.

constraints on the structure or faithfulness of the reasoning process (Guo et al., 2025; Yang et al., 2025; Grattafiori et al., 2024). As a result, models generate reasoning in free-form text, making the resulting traces difficult to verify, analyze, or systematically inspect (Shojaee et al., 2025; Xiong et al., 2025; Baker et al., 2025).

In this work, we address this limitation by constraining the reasoning process to follow a structured, executable format. Rather than generating free-form text, we train models to produce *computational graphs* – directed acyclic graphs in which each node represents an intermediate computation and edges encode dependencies between steps as presented in Figure 1. This representation is particularly natural for *arithmetic reasoning problems* and *claim decomposition problems* (Cobbe et al., 2021; Lai et al., 2017), where solutions can be decomposed into a sequence of well-defined operations with explicit dependencies. By representing such solutions as graphs, each reasoning step becomes explicit and mechanically verifiable: the final answer is obtained by executing the graph rather than by extracting it from unstructured text.

To train models to produce these graphs, we extend Group Relative Policy Optimization (Shao et al., 2024) with reward signals that evaluate not only the correctness of the final answer but also the structural validity of the generated graph. We refer to the resulting training procedure as *Graph-Based GRPO*. Our contributions are summarized as follows:

- We propose *Graph-Based GRPO*, a reinforcement learning method that optimizes structured reasoning by training models to generate computational graphs.
- Using arithmetic and graph-building reasoning tasks, we show that Graph-Based GRPO consistently improves accuracy over Graph SFT across all five datasets and four model families, while reducing generation length by up to $6.1\times$ compared with Default GRPO and up to $3.1\times$ compared with Graph SFT in certain settings.
- We show that graph-based reasoning yields substantially higher faithfulness, indicating that model predictions are derived from the generated computational graphs rather than from implicit internal shortcuts.

Overall, our results suggest that computational graphs provide a promising intermediate representation for training reasoning-capable language models. By explicitly structuring the reasoning process, we obtain models that are more efficient, more faithful, and easier to analyze than those relying on free-form reasoning traces.

2 Related Work

Structured and Graph-Based Reasoning Representations. Chain-of-Thought (CoT) prompting (Wei et al., 2022) elicits intermediate natural-language reasoning, whereas Program-of-Thought (Chen et al., 2022) and Program-aided Language Models (Gao et al., 2023) generate executable programs. However, these representations do not explicitly encode dependencies between intermediate computations. RL-Struct (Hu and Wu, 2025) and Lu et al. (2025) use reinforcement learning to optimize structured outputs, while Tree-of-Thoughts (Yao et al., 2023) and Graph-of-Thoughts (Besta et al., 2024) explore alternative reasoning paths through inference-time search. In contrast, we train models to generate executable computational

graphs using graph-specific reinforcement learning objectives.

An important distinction is the level at which structure is introduced. Program-based methods make computation executable but typically represent it as a sequence of instructions, whereas Tree-of-Thoughts and Graph-of-Thoughts use structured search to explore candidate reasoning paths at inference time. RL-Struct and related approaches optimize the validity of structured outputs, but our objective additionally evaluates intermediate graph content through node-overlap rewards. This allows Graph-Based GRPO to optimize not only output validity but also the reasoning structure used to derive an answer.

Learning to Produce Faithful Reasoning. Reasoning traces may be post-hoc rationalizations rather than faithful accounts of model predictions (Jacovi and Goldberg, 2020; Holzinger et al., 2017; Lanham et al., 2023; Paul et al., 2024; Matton et al., 2025). Prior work addresses this issue through separating statement selection from inference (Creswell and Shanahan, 2022), executing symbolic programs generated from natural-language problems (Lyu et al., 2023), verifying intermediate reasoning steps (Cobbe et al., 2021), or using reinforcement learning to align reasoning chains with final predictions (Paul et al., 2024). Our approach instead uses executable computational graphs and graph-based rewards to encourage compact, mechanically verifiable reasoning.

3 Method

We train models to produce structured reasoning in the form of computational graphs (more about computational graph generation is given in Appendix B, Appendix I and Appendix J) through a two-stage procedure: supervised fine-tuning on graph-annotated data, followed by reinforcement learning with graph-specific reward signals.

Training format. Each sample is a triple (x, g, y) : a natural-language problem x , a gold computational graph g encoding the reasoning process, and the gold answer y .

For the experiments presented in Section 5, the model receives x and generates a graph $\hat{g}$ during both training and inference. For the faithfulness evaluation (Section 6), the model additionally generates an explicit answer ANSWER: $\hat{y}$ following the graph $\hat{g}$ during both training and inference. This

allows us to compare the answer derived from graph execution (internal) with the model’s explicitly generated answer (external) and to conduct intervention-based faithfulness experiments. Entailment is an exception, as it does not require an explicit answer; therefore, internal/external agreement is not evaluated for this dataset.

3.1 Stage 1: Supervised Fine-Tuning

We fine-tune the base model on the (x, g, y) triples of Section 4.2 to generate graphs in the format above. This stage exposes the model to the structured output format and initializes the subsequent RL stage.

3.2 Stage 2: Graph-Based GRPO

Starting from the SFT checkpoint, we optimize the model with GRPO (Shao et al., 2024). For each problem x , the policy π_θ samples K candidates $\{(\hat{g}_1, \hat{y}_1), \dots, (\hat{g}_K, \hat{y}_K)\}$, each scored by a composite reward $R(\hat{g}_i, \hat{y}_i, g, y)$ (Section 3.3). The policy is updated to increase the probability of candidates with positive advantage and decrease it for those with negative advantage. We use $K = 16$; remaining hyperparameters are in Appendix A.

3.3 Reward Design

All rewards share a *style* component that checks structural validity; the remaining components differ by task type. Arithmetic tasks are scored by $S_{\text{main}} = S_s + S_c + S_o$ and proof-construction tasks by $S_{\text{main}} = S_s + S_{\text{leaves}} + S_{\text{steps}}$. For the faithfulness experiments, on datasets with an explicit external answer, we additionally include the answer reward S_a and the faithfulness reward S_f : $S_{\text{faith}} = S_{\text{main}} + S_a + S_f$. The answer reward encourages the explicitly generated answer to match the gold answer, while the faithfulness reward encourages consistency between the explicit answer and the answer obtained by executing the generated graph. Entailment has no external answer and therefore does not use these faithfulness-specific rewards. Together, these components provide informative training signals at different stages: the style reward encourages well-formed graphs, while the remaining rewards evaluate their content, correctness, and, where applicable, answer–graph consistency.

Style reward (all tasks). The output must be a valid JSON list of node dictionaries, optionally followed by an ANSWER: <number/boolean> suffix.

Each node carries an id and a type; const nodes additionally carry a value, and op nodes an op field and an inputs list referencing other node identifiers. Any violation – invalid JSON, non-dictionary elements, or missing and unrecognized fields – is penalized:

$$S_s = \begin{cases} 2.0, & \text{if structurally valid} \\ -2.0, & \text{otherwise.} \end{cases} \quad (1)$$

Correctness reward (arithmetic). We execute the generated graph and compare its value to the gold answer under a graded scheme, so that near-misses are distinguished from failures:

$$S_c = \begin{cases} 3.0, & \text{if } |y_{\text{pred}} - y_{\text{true}}| < \epsilon \\ 1.0, & \text{if } 0.9 \leq y_{\text{pred}}/y_{\text{true}} \leq 1.1 \\ 0.5, & \text{if } 0.8 \leq y_{\text{pred}}/y_{\text{true}} \leq 1.2 \\ -0.5, & \text{if the graph executes incorrectly} \\ -1.5, & \text{if the graph cannot be executed} \\ -2.0, & \text{if no valid graph is found,} \end{cases} \quad (2)$$

with $\epsilon = 10^{-6}$.

Overlap reward (arithmetic). Final-answer correctness alone can be obtained by shortcut strategies, so we additionally score the intermediate steps. We measure the overlap between the values of gold and generated nodes, over both constants and operations:

$$S_o = \begin{cases} |G_c \cap \hat{G}_c| + |G_o \cap \hat{G}_o|, & \text{valid graph,} \\ -1.5, & \text{empty node set,} \\ -2.0, & \text{invalid graph.} \end{cases} \quad (3)$$

where G_c, G_o are the value sets of gold constant and operation nodes and $\hat{G}_c, \hat{G}_o$ the corresponding generated sets.

Leaves reward (proof graphs). We extract the context facts (sent*) referenced in the predicted and gold proofs and compute set-level F1, scaled to $[-1, 1]$:

$$S_{\text{leaves}} = 2 \cdot \text{F1}(L_{\text{pred}}, L_{\text{gold}}) - 1, \quad (4)$$

where L_{pred} and L_{gold} are the sets of used sent identifiers. This rewards selecting the correct supporting facts while ignoring distractors.

Steps reward (proof graphs). Each step is an edge $(lhs, refs)$, where lhs is the output node (int* or hypothesis) and $refs$ is the sorted list of input

Dataset	Train	Test	# Inputs	# Ops
GSM8k+	5,634	1,409	2–4	3–7
iGSM	2,634	658	9–23	5–18
MATH	7,959	1,992	3–5	3–8
Entailment	1,313	340	3–25	1–17
ProofWriter	5,000	2,000	1–5	0–7

Table 1: Datasets used in our experiments. We report train/test sizes and the ranges of input nodes (representing quantities mentioned in the problem, not all of which necessarily appear in the resulting computational graph) and operations (represented by operational nodes in the corresponding computational graphs).

identifiers. We compute F1 over predicted and gold edge sets:

$$S_{\text{steps}} = 3 \cdot \text{F1}(E_{\text{pred}}, E_{\text{gold}}) - 1.5. \quad (5)$$

This rewards correct proof topology – both the number of steps and the dependency structure. Unparseable outputs receive low reward implicitly, via empty predicted sets ($\text{F1} = 0$).

Answer reward (with explicit answer). Correctness scores the executed graph; the answer reward scores only the emitted ANSWER suffix against the gold label:

$$S_a = \begin{cases} 3.0, & \text{if } |y_{\text{ans}} - y_{\text{true}}| < \epsilon \\ -0.5, & \text{if the answer is incorrect} \\ -2.0, & \text{if no answer is found,} \end{cases} \quad (6)$$

with $\epsilon = 10^{-6}$. The model can collect S_a even when the graph is wrong or missing.

Faithfulness reward (with explicit answer). This term requires the parsed answer to match the executed graph, independently of the gold label:

$$S_f = \begin{cases} 3.0, & \text{if } |y_{\text{ans}} - y_{\text{graph}}| < \epsilon \\ -0.5, & \text{if the two values disagree} \\ -1.5, & \text{if the graph cannot be executed} \\ -2.0, & \text{if the answer or graph is missing,} \end{cases} \quad (7)$$

with the same ϵ . Together with S_c and S_a , this rewards a correct executable graph whose emitted answer is that graph’s own result.

4 Experimental Setup

4.1 Models

We evaluate our approach using four instruction-tuned language models from four model families:

Gemma 2 2B, Falcon 3 3B, Llama 3.2 3B, and Qwen3 4B.

4.2 Datasets

We evaluate on five benchmarks: three arithmetic reasoning tasks, where solutions decompose into chains of operations, and two proof-construction tasks, where the graph encodes an entailment structure rather than a computation. Table 1 reports sizes and graph statistics.

GSM8k+ (Cobbe et al., 2021) We build on GSM8k, $\sim 7\text{K}$ grade-school word problems requiring multi-step arithmetic. Because the benchmark is likely present in pre-training corpora (Section 4.3), test performance may partly reflect memorization. We therefore keep each word problem and its computational graph intact but redefine the four arithmetic operators: a script maps each of $+$, $-$, $\times$, $/$ to a per-example linear combination of its two operands. We re-execute the graph under the new semantics, recompute the gold answer, adapt the natural-language solution, and state the operator definitions in the prompt at training and inference time. The reasoning structure is unchanged, but the memorized answer is now wrong (Appendix C).

iGSM (Zeyuan Allen-Zhu, 2023) $\sim 3.3\text{K}$ procedurally generated problems; difficulty scales with the size of the underlying reasoning graph, and many objects and relations are redundant.

MATH (Hendrycks et al., 2021) $\sim 10\text{K}$ high-school competition problems spanning diverse topics and difficulty levels.

Entailment (Dalvi et al., 2021) Expert-annotated grade-school science QA requiring multi-step entailment proofs from facts to a hypothesis. We use Task2, where relevant and distractor facts are both given and most go unused; difficulty grows with proof length.

ProofWriter (Oyvind Tafjord, 2021) True/False questions where each label is justified by a proof over a theory of facts and rules, some of which are distractors. The model must build a path from the relevant premises to the queried statement.

Evaluation. For arithmetic datasets (GSM8K+, iGSM, and MATH), we execute the generated graph $\hat{g}$ in topological order and take the value of its final node as the prediction $\hat{y}$. For free-form baselines, we extract the last number from the generated response. We evaluate predictions using exact match against the gold answer y .

For graph-building datasets (Entailment and

Dataset	Method	Gemma2-2B	LLama3.2-3B	Falcon3-3B	Qwen3-4B	Average
GSM8k+	ICL with FS	0.12 $\pm$ 0.0	0.11 $\pm$ 0.0	0.29 $\pm$ 0.0	0.18 $\pm$ 0.0	0.17 $\pm$ 0.07
	Tree-of-Thoughts	0.00 $\pm$ 0.00	0.01 $\pm$ 0.00	0.02 $\pm$ 0.00	0.12 $\pm$ 0.00	0.04 $\pm$ 0.05
	Answer-only SFT	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00
	CoT + Answer SFT	0.08 $\pm$ 0.00	0.24 $\pm$ 0.00	0.26 $\pm$ 0.00	0.37 $\pm$ 0.01	0.24 $\pm$ 0.10
	Graph SFT	0.24 $\pm$ 0.01	0.31 $\pm$ 0.00	0.33 $\pm$ 0.00	0.43 $\pm$ 0.00	0.33 $\pm$ 0.07
	Default GRPO	0.00 $\pm$ 0.00	0.03 $\pm$ 0.00	0.04 $\pm$ 0.00	0.24 $\pm$ 0.01	0.08 $\pm$ 0.09
	Graph-Based GRPO	0.27 $\pm$ 0.01	0.34 $\pm$ 0.01	0.36 $\pm$ 0.00	0.44 $\pm$ 0.01	0.35 $\pm$ 0.06
iGSM	ICL with FS	0.02 $\pm$ 0.0	0.05 $\pm$ 0.0	0.07 $\pm$ 0.0	0.14 $\pm$ 0.0	0.07 $\pm$ 0.04
	Tree-of-Thoughts	0.29 $\pm$ 0.00	0.22 $\pm$ 0.00	0.29 $\pm$ 0.00	0.49 $\pm$ 0.00	0.32 $\pm$ 0.10
	Answer-only SFT	0.39 $\pm$ 0.01	0.45 $\pm$ 0.01	0.44 $\pm$ 0.01	0.49 $\pm$ 0.01	0.44 $\pm$ 0.04
	CoT + Answer SFT	0.21 $\pm$ 0.01	0.56 $\pm$ 0.01	0.55 $\pm$ 0.01	0.65 $\pm$ 0.0	0.49 $\pm$ 0.17
	Graph SFT	0.45 $\pm$ 0.00	0.45 $\pm$ 0.00	0.43 $\pm$ 0.00	0.55 $\pm$ 0.00	0.47 $\pm$ 0.05
	Default GRPO	0.36 $\pm$ 0.01	0.17 $\pm$ 0.03	0.28 $\pm$ 0.01	0.45 $\pm$ 0.01	0.32 $\pm$ 0.10
	Graph-Based GRPO	0.57 $\pm$ 0.02	0.59 $\pm$ 0.00	0.56 $\pm$ 0.01	0.66 $\pm$ 0.01	0.59 $\pm$ 0.04
MATH	ICL with FS	0.03 $\pm$ 0.0	0.08 $\pm$ 0.0	0.12 $\pm$ 0.0	0.20 $\pm$ 0.0	0.11 $\pm$ 0.06
	Tree-of-Thoughts	0.25 $\pm$ 0.00	0.51 $\pm$ 0.00	0.52 $\pm$ 0.00	0.76 $\pm$ 0.00	0.51 $\pm$ 0.18
	Answer-only SFT	0.04 $\pm$ 0.0	0.32 $\pm$ 0.01	0.23 $\pm$ 0.0	0.43 $\pm$ 0.0	0.26 $\pm$ 0.14
	CoT + Answer SFT	0.10 $\pm$ 0.01	0.51 $\pm$ 0.01	0.40 $\pm$ 0.0	0.59 $\pm$ 0.0	0.40 $\pm$ 0.19
	Graph SFT	0.09 $\pm$ 0.0	0.12 $\pm$ 0.0	0.13 $\pm$ 0.0	0.24 $\pm$ 0.01	0.14 $\pm$ 0.06
	Default GRPO	0.34 $\pm$ 0.01	0.63 $\pm$ 0.01	0.61 $\pm$ 0.01	0.85 $\pm$ 0.00	0.61 $\pm$ 0.18
	Graph-Based GRPO	0.12 $\pm$ 0.00	0.17 $\pm$ 0.01	0.17 $\pm$ 0.01	0.30 $\pm$ 0.00	0.19 $\pm$ 0.07
Entailment	ICL with FS	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.11 $\pm$ 0.00	0.03 $\pm$ 0.05
	Tree-of-Thoughts	0.04 $\pm$ 0.00	0.02 $\pm$ 0.00	0.03 $\pm$ 0.00	0.13 $\pm$ 0.00	0.06 $\pm$ 0.04
	Answer-only SFT	0.00 $\pm$ 0.00	0.01 $\pm$ 0.00	0.01 $\pm$ 0.00	0.10 $\pm$ 0.00	0.03 $\pm$ 0.04
	CoT + Answer SFT	0.02 $\pm$ 0.00	0.04 $\pm$ 0.00	0.02 $\pm$ 0.00	0.10 $\pm$ 0.01	0.04 $\pm$ 0.03
	Graph SFT	0.05 $\pm$ 0.00	0.07 $\pm$ 0.00	0.07 $\pm$ 0.00	0.17 $\pm$ 0.01	0.09 $\pm$ 0.05
	Default GRPO	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00
	Graph-Based GRPO	0.08 $\pm$ 0.03	0.10 $\pm$ 0.04	0.14 $\pm$ 0.02	0.20 $\pm$ 0.01	0.13 $\pm$ 0.05
ProofWriter	ICL with FS	0.05 $\pm$ 0.00	0.07 $\pm$ 0.00	0.25 $\pm$ 0.00	0.40 $\pm$ 0.00	0.19 $\pm$ 0.16
	Tree-of-Thoughts	0.00 $\pm$ 0.00	0.02 $\pm$ 0.00	0.00 $\pm$ 0.00	0.56 $\pm$ 0.00	0.15 $\pm$ 0.28
	Answer-only SFT	0.22 $\pm$ 0.01	0.20 $\pm$ 0.01	0.18 $\pm$ 0.00	0.38 $\pm$ 0.00	0.24 $\pm$ 0.09
	CoT + Answer SFT	0.44 $\pm$ 0.00	0.41 $\pm$ 0.02	0.36 $\pm$ 0.01	0.69 $\pm$ 0.00	0.48 $\pm$ 0.15
	Graph SFT	0.27 $\pm$ 0.00	0.54 $\pm$ 0.01	0.35 $\pm$ 0.00	0.77 $\pm$ 0.01	0.48 $\pm$ 0.22
	Default GRPO	0.04 $\pm$ 0.03	0.17 $\pm$ 0.02	0.04 $\pm$ 0.06	0.13 $\pm$ 0.01	0.10 $\pm$ 0.07
	Graph-Based GRPO	0.51 $\pm$ 0.00	0.58 $\pm$ 0.01	0.53 $\pm$ 0.02	0.81 $\pm$ 0.02	0.61 $\pm$ 0.14

Table 2: Performance with standard deviations across datasets and training methods. Best results are shown in bold and second-best in italic.

ProofWriter), we adopt the Overall AllCorrect metric (Dalvi et al., 2021) from the official Entailment-Bank evaluation framework. This metric requires three components to be simultaneously correct: the sets of leaf nodes and inference steps must exactly match their gold counterparts, while intermediate conclusions must achieve a BLEURT-F1 score of 1. Overall AllCorrect is the product of these three binary indicators, averaged across test examples. We evaluate this metric on proof graphs from Graph SFT and Graph-Based GRPO, and on native proofs from the remaining methods.

For the internal/external faithfulness evaluation (Appendix, Table 10), we instead use answer-level exact match for ProofWriter, independently of the full proof structure. Internal accuracy compares the graph’s final-node result with the gold label; external accuracy compares the model’s explicit True/False answer with the gold label.

Splits. Each dataset is split 80/20 into train and test (71/29 for ProofWriter). For GRPO-Based training the training part is divided equally between SFT and RL.

4.3 Contamination Audit

Strong scores on a public benchmark may reflect pre-training exposure rather than reasoning. We audit all five datasets with CoDeC (Zawalski et al., 2025), which flags a sample as memorized when prepending other samples from the same dataset lowers its likelihood; the dataset score is the fraction of such samples. Against Pythia-1.4B, whose corpus predates both benchmarks - MATH and GSM8k - Qwen3-4B shows scores of 45.9% and 28.2% – three and twenty times the reference – while iGSM, Entailment and ProofWriter stay below 10% (Table 3). This motivates GSM8k+, and it bounds what MATH can show: much of that

Dataset	Qwen3-4B	Pythia-1.4B	Llama-3.2-3B-Instruct	Gemma-2-2B	Falcon3-3B-Instruct
MATH	45.9	14.9	51.2	26.3	70.8
GSM8K	28.2	1.3	34.9	21.2	78.1
GSM8K+	0.0	0.0	0.0	0.0	0.0
iGSM	0.0	0.0	0.0	0.0	1.3
Entailment	9.6	0.0	0.0	0.0	0.3
ProofWriter	0.3	0.0	0.0	0.0	31.1

Table 3: CoDeC contamination scores (%) computed for each benchmark dataset. The score is the fraction of samples with a negative in-context delta $\Delta(x) < 0$; lower values indicate less contamination.

test set is solvable by recall, which attenuates the measured effect of any reasoning method.

4.4 Baselines

We compare our method (Graph SFT + Graph GRPO, Section 3) against six baselines spanning three paradigms. All fine-tuned models use LoRA (Hu et al., 2022); hyperparameters are in Appendix A.

In-context learning. **ICL with few-shot** prompts the base model with two (x, g, y) demonstrations in the graph format and executes the generated $\hat{g}$. **Tree-of-Thoughts** (Yao et al., 2023) is a stronger zero-update baseline that explores a tree of intermediate steps with branching, self-evaluation and backtracking at inference time.

Supervised fine-tuning. **Answer-only SFT** trains on (x, y) pairs and emits an answer with no intermediate reasoning. **CoT + Answer SFT** trains on (x, c, y) triples, where c is the free-form reasoning trace from the original dataset. **Graph SFT** trains on (x, g, y) tuples in our format; it shares our data and output format but omits the GRPO stage, isolating the contribution of supervised graph training.

Reinforcement learning. **Default GRPO** trains free-form reasoning in `<think>` tags with rewards for answer correctness and format compliance, representing the standard RL setup in recent reasoning models.

5 Results

5.1 Main Results

Table 2 and Figure 2 show that Graph-Based GRPO improves over Graph SFT on all five datasets, with the strongest gains on compositional tasks. On GSM8k+ it reaches the best average accuracy (0.35, against 0.33 for Graph SFT and 0.24 for CoT + Answer SFT). The largest improvement is observed on ProofWriter, from 0.48 to 0.61, followed by iGSM, where accuracy increases from 0.47 to 0.59. Since iGSM is procedurally generated to resist memoriza-

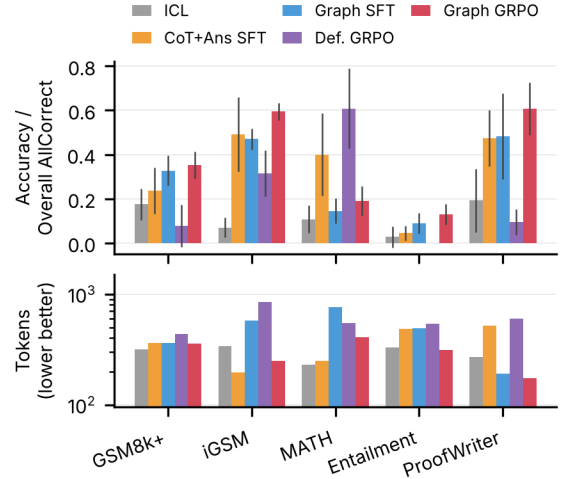
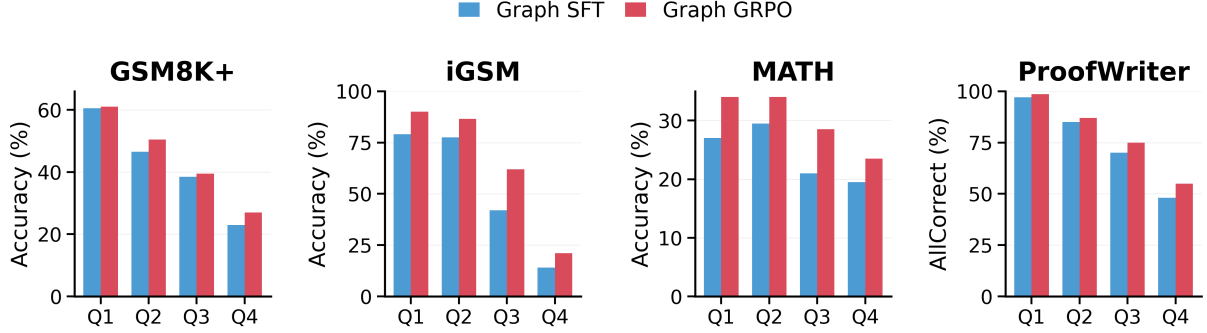


Figure 2: Accuracy/Overall AllCorrect (top) and generation length (bottom, log scale), averaged over the four models; error bars give the spread across models. Graph-Based GRPO leads on four of five datasets at among the lowest token cost; MATH is the exception. Tree-of-Thoughts and Answer-only SFT are omitted for scale and because the latter emits no reasoning. Full results: Tables 2 and 8.

tion, this suggests computational graphs provide a useful inductive bias under distribution shift. On Entailment and ProofWriter, Graph-Based GRPO is again best (0.13 and 0.61), consistent with graph representations aligning naturally with multi-step proof construction.

MATH is the exception: Default GRPO (0.61), Tree-of-Thoughts (0.51), CoT + Answer SFT (0.40) and Answer-only SFT (0.26) substantially outperform Graph-Based GRPO (0.19). Our contamination audit (Section 4.3) offers a partial explanation – MATH is the most contaminated dataset we test, so free-form methods can exploit memorized natural-language solution patterns that structured reasoning does not reproduce.

The lower panel of Figure 2 shows generation length. Graph-Based GRPO produces substantially shorter outputs than Default GRPO while matching



Normalized gold graph length quartile (Q1: shortest, Q4: longest)

Figure 3: Qwen3-4B accuracy (Overall AllCorrect on ProofWriter) stratified by normalized gold graph length. Graph GRPO stays above Graph SFT in all bins and remains more robust on longer graphs.

or exceeding its accuracy on four datasets; the effect is largest on iGSM, where it is both the most accurate and the second cheapest method. Compared with Graph SFT, Graph-Based GRPO generally produces shorter reasoning traces while achieving higher accuracy on all datasets (Appendix E).

Per-model results (Table 2) show that LLaMA3.2-3B, Falcon3-3B and Qwen3-4B benefit more consistently from graph-based RL than Gemma2-2B, suggesting stable structured-reasoning policies require sufficient model capacity.

Across models and datasets, most failures stem from incorrect operations or formulas, indicating that the main challenge is composing correct computations rather than generating graph structure. On GSM8k+, these errors account for 72–87% of failures, while MATH and iGSM show greater variation and more structural errors in some settings. Smaller models, particularly Gemma2-2B, also produce more invalid graphs, highlighting difficulties with structural constraints. Overall, models generally generate valid graph structures but struggle with correct computational composition. More about graph errors at Appendix G.

5.2 Ablation

We ablate each component on Qwen3-4B using GSM8k+ (Table 4). Removing the style reward causes the largest drop (−1.34 points), and removing the correctness reward is nearly as damaging (−1.13). Removing the overlap reward slightly *improves* the mean (+0.12), so on this benchmark structural agreement with the gold graph adds little beyond the other two signals. Replacing the curriculum scheduler with the default scheduler yields

Setting	Mean	Max
Full model	43.88 ± 0.99	45.07
– Style reward	42.54 ± 0.89	43.79
– Correctness reward	42.75 ± 0.65	43.44
– Overlap reward	44.00 ± 1.12	45.49
– Default scheduler	44.00 ± 0.86	45.21
1 rollout	43.88	45.07
8 rollouts + SC	45.56	45.71
16 rollouts + SC	46.98	47.12

Table 4: Ablation study of reward components, scheduling, and the number of rollouts for QWEN3-4B on GSM8k+. Mean accuracy is reported over seeds, with the best result in bold.

a small improvement (+0.12 percentage points). The curriculum scheduler sorts training graphs by increasing number of nodes and edges, training the model from simpler to more complex graphs, whereas the default scheduler randomly shuffles training examples. This suggests that curriculum ordering is not critical for final accuracy in our setting.

Increasing the number of rollouts helps more than any reward change. Moving from one rollout to eight with self-consistency (Wang et al., 2023b) raises accuracy from 43.88% to 45.56%, while sixteen rollouts further improve it to 46.98%, the best result in this ablation.

5.3 Robustness to Graph Length and Sampling

Figure 3 stratifies performance by gold graph length, binned by percentile within each dataset. Across GSM8k+, iGSM, MATH and ProofWriter both methods degrade as graphs grow, but Graph

GRPO stays ahead in every bin, with the clearest gaps at the upper percentiles on iGSM and ProofWriter. RL fine-tuning therefore helps the model sustain multi-step graph reasoning rather than only improving short instances.

On iGSM, Graph GRPO consistently outperforms Graph SFT in pass@ k and scales more reliably with additional samples: while SFT becomes increasingly variable across random seeds as k grows, GRPO maintains stable performance (Appendix F).

6 Faithfulness

Accuracy alone cannot distinguish a model that derives its answer from the generated graph from one that emits a plausible graph and answers by another route. We assess this in two ways, summarized in Table 5.

Internal–external agreement. We score each model twice: *internally*, by executing the generated graph and taking its final node, and *externally*, by reading the answer the model emits in its ANSWER field. A small gap Δ between the two is necessary – though not sufficient – for the graph to be the source of the answer.

On iGSM and MATH the gap nearly vanishes after GRPO (1.0 $\rightarrow$ 0.2 and 1.7 $\rightarrow$ 0.4 points), and it falls by a factor of three on ProofWriter. GSM8k+ is the exception: the gap is large for every method and *widens* under training, from 12.1 for ICL to 31.9 for Graph GRPO. The cause is visible in the per-model numbers: for Qwen3-4B, internal accuracy is 44.29% while external accuracy against the redefined-operator gold is 0.20% – yet external accuracy against the *original* GSM8k answers is 42.0%, almost matching the internal score. Despite the redefined operators in the prompt, the model still emits the memorized answer, consistent with the contamination audit in Section 4.3.

Intervention-based faithfulness. Agreement is only correlational, so we additionally follow the intervention protocol of Somov et al. (2026), which treats the generated graph as a mediator M with a deterministic executor $C(\cdot)$ mapping any graph to the answer it implies. F_{ID} asks whether the model’s answer matches $C(\hat{M})$ for its own graph; F_{OOD} asks whether, after editing the graph ($\hat{M} \rightarrow M^*$ by some kind of structural edit – node or operation replacement) and re-prompting, the new answer should also match $C(M^*)$; F_{strong} requires both

Method	GSM8K+	iGSM	MATH	PW
Δ internal–external (pp, lower better)				
ICL	12.1 ₃	21.1 ₁₈	25.2 ₃₀	14.3 ₅
Graph SFT	28.4 ₉	1.0 ₀	1.7 ₁	11.8 ₅
Graph GRPO	31.9 ₉	0.2 ₀	0.4 ₀	3.5 ₁
F_{strong} (% , higher better)				
ICL	0.0 ₀	2.1 ₁	1.7 ₁	28.8 ₂₅
Graph SFT	0.0 ₀	44.0 ₆	21.4 ₆	49.4 ₂₂
Graph GRPO	0.0 ₀	48.2 ₃	40.2 ₂	52.4 ₂₂

Table 5: Faithfulness, averaged over the four models; subscripts give the spread across models, rounded to the nearest point. Δ is the absolute gap between internal (graph execution) and external (emitted answer) accuracy; F_{strong} requires answer–graph consistency both before and after intervention. **Bold** = best per column. Entailment has no external answer; therefore, neither internal–external accuracy nor answer-based faithfulness metrics are applicable. Per-model values, and F_{ID}/F_{OOD} , are in Table 10.

F_{ID} and F_{OOD} to be true. We report F_{strong} , the strictest of the three in Table 5.

Graph GRPO raises F_{strong} on every dataset where the metric is meaningful: MATH nearly doubles (21.4 $\rightarrow$ 40.2), and iGSM and ProofWriter improve from already-high baselines (44.0 $\rightarrow$ 48.2 and 49.4 $\rightarrow$ 52.4). Editing the graph therefore changes the answer more reliably after RL than after supervised training alone.

GSM8k+ is again the outlier: F_{strong} is exactly 0 for both graph-based methods across all four models. Together with the widening internal–external gap, this suggests the model can construct a well-formed graph while relying on a separate memorized pathway for its answer.

7 Conclusion

We have explored structured reasoning as an alternative to free-form chain-of-thought by training language models to generate executable computational graphs. Our experiments show that reinforcement learning over graph-structured reasoning improves accuracy and efficiency, with Graph-Based GRPO consistently outperforming supervised graph training, particularly on compositional tasks, while producing shorter reasoning traces. Intervention experiments further reveal higher faithfulness, as modifying the reasoning graph changes the final answer, indicating genuine reliance on the generated structure. Overall, executable intermediate structures offer a promising path toward more reliable and verifiable reasoning in LLMs.

Limitations

While our Graph-Based GRPO method demonstrates consistent improvements over Graph SFT, it still underperforms compared to CoT + Answer SFT and Default GRPO in some cases. This suggests that, despite the advantages of structured reasoning, there remains substantial room for improvement, particularly in the design of reward functions and output representations, as highlighted by our ablation study.

Our method is limited to tasks whose reasoning processes can be naturally represented as computational graphs. While this representation is suitable for mathematical reasoning and structured question answering, it may be less applicable to tasks that require open-ended, qualitative, or primarily linguistic reasoning that cannot be readily expressed as explicit computational dependencies.

References

- Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. 2025. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint arXiv:2503.11926*.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690.
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Antonia Creswell and Murray Shanahan. 2022. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*.
- Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. *arXiv preprint arXiv:2104.08661*.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International conference on machine learning*, pages 10764–10799. PMLR.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Yunseok Han, Yejoon Lee, and Jaeyoung Do. 2026. Rfeval: Benchmarking reasoning faithfulness under counterfactual reasoning intervention in large reasoning models. *arXiv preprint arXiv:2602.17053*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- Andreas Holzinger, Chris Biemann, Constantinos S Patsichis, and Douglas B Kell. 2017. What do we need to build explainable ai systems for the medical domain? *arXiv preprint arXiv:1712.09923*.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.
- Ruike Hu and Shulei Wu. 2025. RL-struct: A lightweight reinforcement learning framework for reliable structured output in llms. *arXiv preprint arXiv:2512.00319*.
- Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4198–4205.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.
- Alice Lai, Yonatan Bisk, and Julia Hockenmaier. 2017. Natural language inference from multiple premises. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 100–109.

- Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamile Lukosiute, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, and 11 others. 2023. [Measuring faithfulness in chain-of-thought reasoning](#). *CoRR*, abs/2307.13702.
- Yaxi Lu, Haolun Li, Xin Cong, Zhong Zhang, Yesai Wu, Yankai Lin, Zhiyuan Liu, Fangming Liu, and Maosong Sun. 2025. Learning to generate structured output with schema reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4905–4918.
- Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 305–329.
- Katie Matton, Robert Osazuwa Ness, John Guttag, and Emre Kıcıman. 2025. Walk the talk? measuring the faithfulness of large language model explanations. *arXiv preprint arXiv:2504.14150*.
- Peter Clark Oyvind Taffjord, Bhavana Dalvi Mishra. 2021. [Proofwriter: Generating implications, proofs, and abductive statements over natural language](#). *arXiv preprint arXiv:2012.13048*.
- Debjit Paul, Robert West, Antoine Bosselut, and Boi Faltings. 2024. Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15012–15032.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Parshin Shojaei, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *SuperIntelligence-Robotics-Safety & Alignment*, 2(6).
- Oleg Somov, Mikhail Chaichuk, Gleb Ershov, Karim Vafin, Mikhail Seleznyov, Alexander Panchenko, and Elena Tutubalina. 2026. [Breaking the chain: A causal analysis of llm faithfulness to intermediate structures](#). *arXiv preprint arXiv:2603.16475*.
- Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. 2023. Language models don’t always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965.
- Martin Tutek, Fateme Hashemi Chaleshtori, Ana Marasović, and Yonatan Belinkov. 2025. Measuring chain of thought faithfulness by unlearning reasoning steps. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9946–9971.
- Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023a. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2717–2739.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. [Self-consistency improves chain of thought reasoning in language models](#). *arXiv preprint arXiv:2203.11171*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits its reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Zidi Xiong, Shan Chen, Zhenting Qi, and Himabindu Lakkaraju. 2025. Measuring the faithfulness of thinking drafts in large reasoning models. *arXiv preprint arXiv:2505.13774*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.
- Michał Zawalski, Meriem Boubdir, Klaudia Bałazy, Be-smira Nushi, and Pablo Ribalta. 2025. Detecting data contamination in llms via in-context learning. *arXiv preprint arXiv:2510.27055*.
- Yuanzhi Li Zeyuan Allen-Zhu. 2023. [Physics of language models](#). *Computing Research Repository*, arXiv:2309.14402. Version 2.

A Model’s Hyperparameters and Training Methodology

All models were trained with LoRA, 3 epochs for SFT pipelines, 600 iterations of GRPO for default GRPO pipeline and 1 epoch for pre-GRPO SFT checkpoint and 600 iterations for Graph GRPO

pipeline. Models trained for 3 seed runs - 0, 42, 2026.

On average, SFT stages (answer-only, CoT+answer, and Graph SFT) take about 20 minutes per seed, while GRPO stages (default GRPO and Graph GRPO) take about 2.5 hours per seed. All runs used 4×A100 GPUs with 80 GB memory.

LoRA $\alpha = 32, r = 32$, attention (q_proj, k_proj, v_proj, o_proj) and mlp (gate_proj, up_proj, down_proj) modules are tuned.

GRPO num_generations=16 for Default GRPO and for Graph GRPO pipelines.

Other lr= 5×10^{-6} for SFT and GRPO pipelines, lr= 3×10^{-4} for pre-GRPO checkpoint, paged_adamw_8bit optimizer (Kingma and Ba, 2014), batch_size=2, gradient_accumulation=2.

B Computational graph generation algorithm

To generate computational graphs for GSM8K+, iGSM and MATH, we use a few-shot prompting strategy. Specifically, two example problems from the same domain are included in the system prompt to demonstrate the desired graph format and guide the model toward producing structured outputs.

We use a low sampling temperature ($T = 0.2$) to ensure stable graph generation and reduce variability in the produced structures. All graphs are generated using GPT-4o.

For arithmetical datasets, to ensure the validity of the generated graphs, we execute a forward pass through each graph and compare the computed answer with the ground-truth solution. We additionally enforce that the final node of the graph must be an operation node rather than a constant node. This constraint prevents degenerate cases in which the model outputs the correct answer directly while the preceding nodes and edges do not contribute to the final computation. The graph generation algorithm is identical for all three mathematical datasets.

For the graph-building datasets, Entailment and ProofWriter, we reconstruct computational graphs from the original solutions.

The prompt is in Appendix I.

C GSM8k+ building

C.1 Motivation

GSM8k is a standard benchmark, but strong scores on the public test set may reflect pre-training exposure rather than genuine reasoning. We therefore reconstruct the benchmark to reduce the risk that a model simply recalls familiar question-answer pairs from the original release.

C.2 What we kept and what we changed

We keep the original word problem and the same graph structure: the same nodes and the same assignment lines (e.g., ryan=karl+20, ben=ryan-10). Only the semantics of the four operators, +, -, *, and /, are changed. Using a script, each example receives its own mapping from these symbols to linear combinations of the two operands (e.g., + may be defined as $1 \cdot a + 1 \cdot b - 2$). The graph is then evaluated with these redefined operations, and the resulting value is used as the graph-level gold answer. We also modify our NLU solutions to account for the redefined operations. In addition, we add the new operator definitions to each prompt during both training and inference.

C.3 Why this helps against contamination

Because the underlying data structure is changed, a model cannot solve the task by memorizing the numbers from the original benchmark.

D Contamination Analysis of the Benchmark Datasets

A reliable comparison of different reasoning strategies in LLMs requires verifying that the benchmark datasets used for evaluation have not leaked into the models’ pretraining corpora. Data contamination inflates the reported metrics and confounds the measured effect of the reasoning procedure itself, because a model that has memorized the evaluation data solves the tasks by retrieving memorized patterns rather than by generalizing through the reasoning process. We therefore perform a systematic contamination audit of all datasets considered in this work.

D.1 Detection Method

We detect contamination using CoDeC (Contamination Detection via Context) (Zawalski et al., 2025), a model- and dataset-agnostic method that requires only gray-box access to token probabilities. For each sample x in a candidate dataset $\mathcal{D}$,

CoDeC compares two quantities: (i) the *baseline confidence*, defined as the average log-likelihood of the tokens of x scored in isolation, and (ii) the *in-context confidence*, obtained by prepending n other samples drawn from $\mathcal{D} \setminus \{x\}$ to x and measuring the average log-likelihood of the tokens of x in this augmented context. The contamination signal of a sample is the difference

$$\Delta(x) = \log p_{ic}(x) - \log p_{base}(x), \quad (8)$$

averaged over several context draws. For unseen data, additional in-context examples typically increase the model’s confidence ($\Delta(x) > 0$), whereas for memorized data the context disrupts memorization and reduces confidence ($\Delta(x) < 0$). The dataset-level contamination score is defined as the fraction of samples with $\Delta(x) < 0$.

Following the recommendations of (Zawalski et al., 2025), we use a single in-context example ($n = 1$), average the per-sample delta over 5 random context draws, and exclude the first 10 predicted tokens of the target when computing the average log-likelihood; input sequences are truncated to at most 4096 tokens.

D.2 Contamination Scores

Table 3 reports the CoDeC scores computed for four models—Qwen3-4B, Llama-3.2-3B-Instruct, Gemma-2-2B, and Falcon3-3B-Instruct—together with a reference model, Pythia-1.4B (Section D.3). For MATH and GSM8K the fraction of samples with a negative delta is elevated across all four models, ranging from 26.3% (Gemma-2-2B) to 70.8% (Falcon3-3B-Instruct) on MATH and from 21.2% (Gemma-2-2B) to 78.1% (Falcon3-3B-Instruct) on GSM8K. By contrast, the scores for iGSM and Entailment remain below 10% for all models. For ProofWriter, the scores are near zero for Qwen3-4B, Llama-3.2-3B-Instruct, and Gemma-2-2B, whereas Falcon3-3B-Instruct reaches 31.1%.

We also audit the redefined-operator variant GSM8K+. This dataset is derived from the exact same underlying problems as the original benchmark: the wording, numbers, format, and difficulty are unchanged, and only the operator definitions (and the resulting solutions) are replaced, so that the resulting texts could not have appeared in any pretraining corpus. The contamination score of GSM8K+ is 0.0% for every model (Table 3), with strictly positive per-sample deltas (mean $\Delta \approx +1.1$ to $+1.4$). Because the redefined version holds ev-

ery property of the original benchmark constant except the specific, memorizable text, its zero score rules out any explanation of the original benchmark’s elevated scores based on format, wording, or task difficulty. The only remaining explanation for the discrepancy is that the specific text of the original GSM8K was memorized, which provides an additional, within-model confirmation that this benchmark partially leaked into the models’ pre-training data.

D.3 Statistical Comparison against a Reference Model

To establish that the elevated scores of MATH and GSM8K reflect genuine memorization rather than dataset-specific properties (e.g., an intrinsically low baseline confidence), we follow the reference-based comparison recommended by (Zawalski et al., 2025) and contrast the scores of each target model with those of a reference model known to be uncontaminated. We use Pythia-1.4B, which was pretrained on The Pile, a corpus whose composition was frozen in 2021, before the release of MATH and GSM8K; neither benchmark could therefore appear in its training data.

For each pair of per-sample Δ distributions we conduct two one-sided tests. The *Mann–Whitney U test* (Wilcoxon rank-sum test) is applied to the per-sample Δ values, with the alternative hypothesis that the deltas of the target model are stochastically smaller (more negative) than those of the reference; the magnitude of the effect is quantified by Cliff’s δ . The *Fisher’s exact test* is applied to the 2×2 contingency table of negative ($\Delta < 0$) versus non-negative ($\Delta \geq 0$) samples, with the alternative hypothesis that the odds of a negative delta are higher in the target model than in the reference. We additionally report 95% Wilson score intervals for both contamination proportions.

Table 6 summarizes the results for MATH and GSM8K.

D.4 Comparison of Original and Redefined-Operator Datasets

To quantify this additional evidence of memorization, we contrast the original GSM8K with its redefined-operator counterpart GSM8K+ evaluated on the same model. Table 7 reports the two-sample comparison between the original dataset (target) and the redefined dataset (reference). For every model the original dataset has a strictly lower per-sample delta distribution than the redefined

Model	Score (%)	Wilson 95% CI (%)	Mann–Whitney U			Fisher’s exact	
	Model / Pythia	Model / Pythia	U	p	Cliff’s δ	OR	p
<i>MATH</i>							
Qwen3-4B	45.9 / 14.9	43.7–48.1 / 13.4–16.6	1 403 460	$< 10^{-6}$	0.276	4.831	$< 10^{-6}$
Llama-3.2-3B-Instruct	51.2 / 14.8	50.1–52.3 / 14.1–15.6	15 448 768.5	$< 10^{-6}$	0.498	6.016	$< 10^{-6}$
Gemma-2-2B	26.3 / 14.8	25.3–27.3 / 14.1–15.6	17 161 310	$< 10^{-6}$	0.443	2.045	$< 10^{-6}$
Falcon3-3B-Instruct	70.8 / 14.8	69.8–71.8 / 14.1–15.6	10 488 688.5	$< 10^{-6}$	0.660	13.901	$< 10^{-6}$
<i>GSM8K</i>							
Qwen3-4B	28.2 / 1.3	25.9–30.6 / 0.8–2.0	940 049	2.5×10^{-3}	0.061	30.479	$< 10^{-6}$
Llama-3.2-3B-Instruct	34.9 / 1.3	32.5–37.4 / 0.8–2.0	168 582.5	$< 10^{-6}$	0.832	41.629	$< 10^{-6}$
Gemma-2-2B	21.2 / 1.3	19.2–23.4 / 0.8–2.0	191 762.5	$< 10^{-6}$	0.809	20.882	$< 10^{-6}$
Falcon3-3B-Instruct	78.1 / 1.3	75.9–80.2 / 0.8–2.0	53 643	$< 10^{-6}$	0.946	276.646	$< 10^{-6}$

Table 6: Two-sample comparison of the per-sample CoDeC deltas between each target model and Pythia-1.4B (reference). Values are reported as target/reference. The Mann–Whitney U test is one-sided (H_1 : target deltas stochastically smaller) and Fisher’s exact test is one-sided (H_1 : higher odds of a negative delta in the target); Cliff’s δ is the effect size for the rank test.

dataset: the Mann–Whitney U test is significant at $p < 10^{-6}$ in all cases, Cliff’s δ ranges from 0.97 to 1.00, and Fisher’s exact test yields an infinite odds ratio because the redefined dataset contains no negative deltas at all. The difference between the original and the redefined dataset is therefore not only large but also statistically unambiguous.

D.5 Discussion

For MATH, the contamination scores of all four target models exceed that of Pythia-1.4B ($\approx 14.8\%$), and the differences are statistically significant (Mann–Whitney $p < 10^{-6}$; Fisher’s exact $p < 10^{-6}$): Qwen3-4B reaches 45.9%, Llama-3.2-3B-Instruct 51.2%, Gemma-2-2B 26.3%, and Falcon3-3B-Instruct 70.8%. For GSM8K the discrepancy is even more pronounced: all target models score far above Pythia-1.4B (1.3%), with Falcon3-3B-Instruct reaching 78.1% and Fisher’s odds ratios of up to ≈ 277 . These results provide strong evidence that MATH and GSM8K were partially present in the pretraining data of all four models, most severely in Falcon3-3B-Instruct. In addition, Falcon3-3B-Instruct shows a notable ProofWriter score (31.1%), suggesting partial contamination of that dataset as well.

This contamination may explain why the reasoning approaches introduced in this paper yield comparatively modest improvements on MATH and GSM8K: a substantial fraction of the evaluation examples are solved through memorization rather than through the reasoning process, which attenuates the measured effect of the proposed methods. Finally, the clean 0.0% score of GSM8K+ across all models reinforces this conclusion: this dataset

retains the exact wording, format, and difficulty of the original benchmark and differs only in the redefined operators, yet it produces no contamination signal at all. The fact that a minimal, non-memorizable textual change is sufficient to erase the contamination signal shows that the elevated scores of the original GSM8K are due to memorization of its specific text, providing further evidence that it partially leaked into the models’ pretraining data.

E Token consumption

Table 8 reports the average number of tokens generated per solution across datasets, models, and training strategies. This analysis provides additional insight into the efficiency of different reasoning approaches.

Natural language reasoning vs structured reasoning. Graph-Based GRPO produces shorter outputs than Default GRPO in almost every setting. The gap is modest on GSM8k+ and MATH but grows to more than threefold on iGSM and ProofWriter, the tasks with the longest reasoning chains, and reaches about $6\times$ for Qwen3-4B on iGSM. Default GRPO rewards only answer correctness and format, so its free-form traces tend to grow during training. Graph-Based GRPO writes each step as a compact node with explicit inputs, which leaves little room for verbosity. These savings come with higher accuracy on four of the five datasets. Two out of three cases where Default GRPO is shorter coincide with near-zero accuracy. The advantage also extends beyond arithmetic. On the proof-construction tasks, Entail-

Table 7: Two-sample comparison of the original GSM8K versus its redefined-operator version (GSM8K+) on the same model. Values are reported as original / redefined. The Mann–Whitney U test is one-sided (H_1 : original deltas stochastically smaller) and Fisher’s exact test is one-sided (H_1 : higher odds of a negative delta in the original); Cliff’s δ is the effect size for the rank test.

Model	Dataset	Score (%)	Wilson 95% CI (%)	Mann–Whitney U			Fisher’s exact	
		Orig. / Redef.	Orig. / Redef.	U	p	Cliff’s δ	OR	p
Qwen3-4B	GSM8K	28.2 / 0.0	25.9–30.6 / 0.0–0.3	25 530.0	$< 10^{-6}$	0.974	∞	$< 10^{-6}$
Llama-3.2-3B-Instruct	GSM8K	34.9 / 0.0	32.5–37.4 / 0.0–0.3	0.0	$< 10^{-6}$	1.000	∞	$< 10^{-6}$
Gemma-2-2B	GSM8K	21.2 / 0.0	19.2–23.4 / 0.0–0.3	0.0	$< 10^{-6}$	1.000	∞	$< 10^{-6}$
Falcon3-3B-Instruct	GSM8K	78.1 / 0.0	75.9–80.2 / 0.0–0.3	0.0	$< 10^{-6}$	1.000	∞	$< 10^{-6}$

ment and ProofWriter, Graph-Based GRPO is both shorter and more accurate than CoT + Answer SFT for every model.

Effect of reinforcement learning on generation length. One of the most notable trends in Table 8 is that Graph-Based GRPO often reduces generation length relative to Graph SFT, although the effect depends on the model and dataset. Substantial reductions occur on iGSM, particularly for Falcon3-3B (788 to 251 tokens) and Gemma2-2B (829 to 303 tokens). Similar reductions can be observed for Qwen3-4B on iGSM (248 $\rightarrow$ 233 tokens) and MATH (634 $\rightarrow$ 365 tokens). However, generation length increases in some settings, including Qwen3-4B on GSM8k+ (428 to 478 tokens) and Falcon3-3B on ProofWriter (158 to 221 tokens).

This suggests that reinforcement learning not only improves reasoning accuracy but also encourages the model to generate more concise computational graphs. Intuitively, GRPO rewards solutions that lead to correct answers, which implicitly discourages unnecessary intermediate nodes and redundant operations.

Comparison with Default GRPO. The difference between Default GRPO and Graph-Based GRPO is particularly striking in terms of token efficiency. Default GRPO often produces very long outputs, especially on more difficult datasets such as iGSM. For example, Default GRPO with Falcon3-3B generates on average 840 tokens on iGSM, compared to only 251 tokens for Graph-Based GRPO. A similar pattern can be observed for Qwen3-4B (1423 vs 233 tokens).

These results indicate that graph-based reasoning provides a much more compact representation of the reasoning process than natural language chain-of-thought. Instead of describing intermediate steps in natural language, the model directly produces executable reasoning graphs.

Gemma token anomaly. An interesting anomaly can be observed for the Gemma2-2B model. Despite being the smallest model in our study (2B parameters), it often produces unusually long outputs, particularly when trained with CoT-based methods. For example, on GSM8k+ the CoT + Answer SFT approach generates more than 860 tokens on average, significantly exceeding the generation length of larger models such as Llama3.2-3B or Falcon3-3B.

Manual inspection suggests that this behavior is frequently associated with hallucinated reasoning traces, where the model generates excessively verbose or redundant intermediate steps that do not contribute to the final computation. This phenomenon is consistent with previous observations that smaller models tend to rely on longer and less structured reasoning traces when attempting to solve complex tasks.

Summary. Overall, the results show that graph-based reasoning significantly reduces token consumption while maintaining or improving performance. Reinforcement learning further enhances this effect by encouraging more compact reasoning graphs. These findings suggest that computational graphs provide a more efficient intermediate representation for reasoning compared to natural language explanations.

F Multi-seed pass@ k evaluation.

Figure 4 reports pass@ k on iGSM across three random seeds, with shaded regions denoting 95% confidence intervals. Graph GRPO consistently outperforms Graph SFT across all k values. Moreover, its confidence interval remains relatively stable as k increases, whereas SFT exhibits greater seed-level variability. This suggests that Graph GRPO benefits from additional samples without substantially increasing sensitivity to the training

Dataset	Method	Gemma2-2B	LLama3.2-3B	Falcon3-3B	Qwen3-4B
GSM8k+	ICL with FS	105 $\pm$ 46	188 $\pm$ 173	86 $\pm$ 40	880 $\pm$ 312
	Tree-of-Thoughts	2476 $\pm$ 396	4442 $\pm$ 594	4868 $\pm$ 775	6907 $\pm$ 1031
	Answer-only SFT	8 $\pm$ 2	4 $\pm$ 10	12 $\pm$ 86	3 $\pm$ 1
	CoT + Answer SFT	869 $\pm$ 213	112 $\pm$ 92	251 $\pm$ 149	216 $\pm$ 130
	Graph SFT	348 $\pm$ 4	267 $\pm$ 2	401 $\pm$ 7	428 $\pm$ 27
	Default GRPO	216 $\pm$ 4	364 $\pm$ 28	504 $\pm$ 35	669 $\pm$ 89
	Graph-Based GRPO	324 $\pm$ 3	264 $\pm$ 3	359 $\pm$ 2	478 $\pm$ 259
iGSM	ICL with FS	326 $\pm$ 118	361 $\pm$ 161	218 $\pm$ 136	445 $\pm$ 137
	Tree-of-Thoughts	2476 $\pm$ 414	5072 $\pm$ 740	4632 $\pm$ 1220	5955 $\pm$ 2109
	Answer-only SFT	5 $\pm$ 1	4 $\pm$ 1	5 $\pm$ 0	5 $\pm$ 1
	CoT + Answer SFT	410 $\pm$ 167	127 $\pm$ 145	131 $\pm$ 150	112 $\pm$ 122
	Graph SFT	829 $\pm$ 30	448 $\pm$ 20	788 $\pm$ 17	248 $\pm$ 25
	Default GRPO	317 $\pm$ 47	832 $\pm$ 23	840 $\pm$ 18	1423 $\pm$ 15
	Graph-Based GRPO	303 $\pm$ 23	216 $\pm$ 15	251 $\pm$ 9	233 $\pm$ 16
MATH	ICL with FS	279 $\pm$ 124	165 $\pm$ 177	92 $\pm$ 90	383 $\pm$ 168
	Tree-of-Thoughts	2476 $\pm$ 458	3817 $\pm$ 1068	3878 $\pm$ 1436	5213 $\pm$ 2034
	Answer-only SFT	5 $\pm$ 6	4 $\pm$ 0	6 $\pm$ 6	3 $\pm$ 1
	CoT + Answer SFT	507 $\pm$ 63	150 $\pm$ 130	189 $\pm$ 157	154 $\pm$ 137
	Graph SFT	823 $\pm$ 68	580 $\pm$ 7	1000 $\pm$ 13	634 $\pm$ 19
	Default GRPO	349 $\pm$ 13	404 $\pm$ 5	633 $\pm$ 24	795 $\pm$ 4
	Graph-Based GRPO	341 $\pm$ 13	422 $\pm$ 26	498 $\pm$ 53	365 $\pm$ 7
Entailment	ICL with FS	372 $\pm$ 88	325 $\pm$ 72	156 $\pm$ 50	458 $\pm$ 51
	Tree-of-Thoughts	2754 $\pm$ 597	4588 $\pm$ 587	4277 $\pm$ 822	4134 $\pm$ 606
	Answer-only SFT	132 $\pm$ 219	105 $\pm$ 223	113 $\pm$ 159	118 $\pm$ 231
	CoT + Answer SFT	712 $\pm$ 395	358 $\pm$ 377	572 $\pm$ 419	306 $\pm$ 319
	Graph SFT	479 $\pm$ 13	502 $\pm$ 30	499 $\pm$ 25	482 $\pm$ 9
	Default GRPO	194 $\pm$ 34	786 $\pm$ 0	660 $\pm$ 5	517 $\pm$ 169
	Graph-Based GRPO	386 $\pm$ 127	305 $\pm$ 66	297 $\pm$ 19	259 $\pm$ 18
ProofWriter	ICL with FS	163 $\pm$ 46	237 $\pm$ 61	551 $\pm$ 179	128 $\pm$ 39
	Tree-of-Thoughts	2073 $\pm$ 394	4132 $\pm$ 520	2717 $\pm$ 447	5242 $\pm$ 677
	Answer-only SFT	29 $\pm$ 1	27 $\pm$ 1	31 $\pm$ 2	34 $\pm$ 2
	CoT + Answer SFT	459 $\pm$ 11	846 $\pm$ 8	425 $\pm$ 19	354 $\pm$ 5
	Graph SFT	225 $\pm$ 24	233 $\pm$ 4	158 $\pm$ 4	146 $\pm$ 1
	Default GRPO	446 $\pm$ 90	727 $\pm$ 123	694 $\pm$ 235	533 $\pm$ 96
	Graph-Based GRPO	205 $\pm$ 37	144 $\pm$ 19	221 $\pm$ 26	124 $\pm$ 5

Table 8: Token consumption comparison per generation across datasets and training methods. Standard deviations are shown in smaller font for readability.

seed.

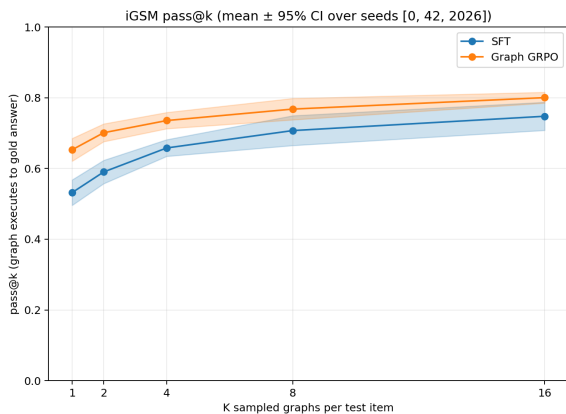


Figure 4: Pass@ k on iGSM across three random seeds. Shaded regions indicate 95% confidence intervals over seeds. Graph GRPO consistently outperforms SFT and exhibits more stable performance as the number of sampled graphs increases.

G Error research

Table 9 presents a detailed breakdown of error categories across datasets and models. Several trends can be observed from the table.

Dominance of reasoning errors. Across most models and datasets, the largest proportion of failures is caused by incorrect operations or formulas. This indicates that the primary difficulty lies in selecting and composing the correct arithmetic operations rather than generating the graph structure itself.

Dataset difficulty effects. The distribution of error types varies across datasets. On GSM8k+, incorrect operations and formulas dominate, accounting for 72–87% of errors. A similar pattern is observed on MATH and iGSM, although these datasets exhibit greater variation in error distribu-

Dataset	Model	Not valid graphs	Not initialized nodes	Unused nodes	Incorrect ops/formulas
GSM8k+	Falcon3-3B	2 ± 0	2 ± 0	18 ± 2	78 ± 2
	Gemma2-2B	7 ± 1	2 ± 0	20 ± 2	72 ± 1
	LLama3.2-3B	4 ± 1	1 ± 0	19 ± 3	75 ± 2
	Qwen3-4B	2 ± 0	1 ± 0	9 ± 1	87 ± 1
iGSM	Falcon3-3B	1 ± 0	2 ± 1	13 ± 4	85 ± 3
	Gemma2-2B	94 ± 3	2 ± 2	2 ± 1	2 ± 2
	LLama3.2-3B	1 ± 0	2 ± 1	26 ± 9	71 ± 10
	Qwen3-4B	2 ± 1	2 ± 1	36 ± 3	60 ± 3
MATH	Falcon3-3B	1 ± 0	5 ± 0	18 ± 2	76 ± 2
	Gemma2-2B	52 ± 32	13 ± 7	10 ± 5	26 ± 33
	LLama3.2-3B	2 ± 1	3 ± 0	23 ± 1	71 ± 1
	Qwen3-4B	3 ± 1	5 ± 0	19 ± 2	73 ± 2

Table 9: Distribution of error categories (%) for Graph-Based GRPO across datasets and models (mean and standard deviation over 3 seeds). Incorrectly initialized constant nodes are counted under *Incorrect ops/formulas*.

tions, with a higher proportion of structural errors in some settings.

Model size effects. Smaller models struggle more with producing valid structured outputs. In particular, Gemma2-2B exhibits a very high rate of invalid graphs on iGSM and MATH, suggesting that maintaining strict structural constraints is difficult for smaller models.

Unused and missing nodes. Errors related to unused nodes and missing node initialization occur when the model partially decomposes the problem but fails to correctly connect intermediate steps. These errors highlight the difficulty of maintaining consistent dependency structures within the generated graphs.

Overall, these results suggest that while models are generally capable of generating the correct graph format, the main challenge remains learning correct computational compositions within the graph structure.

H Faithfulness: Full Results

Dataset	Method	Gemma2-2B	LLama3.2-3B	Falcon3-3B	Qwen3-4B
<i>(a) Internal (graph execution) / External (emitted answer) accuracy (%).</i>					
GSM8K+	ICL	9.23/ 0.14	10.93/0.07	19.02/ 1.06	10.79/ 21.22
	Graph SFT	19.33/0.09	25.34/ 0.38	27.06/0.26	43.20/0.51
	Graph-Based GRPO	19.92/0.14	29.88/0.24	34.47/0.31	44.29/0.20
iGSM	ICL	2.58/15.81	1.22/12.01	1.98/10.94	5.17/56.53
	Graph SFT	45.19/46.40	44.83/46.50	43.26/43.97	55.43/55.12
	Graph-Based GRPO	48.48/48.63	50.00/50.25	47.42/47.37	67.93/67.63
MATH	ICL	2.56/9.74	6.07/7.58	4.92/ 21.69	13.65/ 89.06
	Graph SFT	7.81/8.65	11.78/ 15.29	12.17/13.25	17.12/18.31
	Graph-Based GRPO	12.14/12.44	13.52/12.83	9.47/9.50	29.37/29.82
ProofWriter	ICL	56.00/65.45	25.20/4.75	28.50/46.10	82.75/92.35
	Graph SFT	70.75/90.87	88.08/94.80	81.75/ 94.90	86.88/94.01
	Graph-Based GRPO	91.18/95.78	93.12/97.90	90.13/93.43	98.91/97.76
<i>(b) F_{ID} / F_{OOD} / F_{strong} (%).</i>					
GSM8K+	ICL	0.29/0.36/0.00	1.06/0.18/0.00	1.61/0.21/0.00	25.12/0.16/0.00
	Graph SFT	0.17/0.14/0.00	0.43/ 0.36/0.00	0.14/0.12/0.00	0.58/0.24/0.00
	Graph-Based GRPO	0.14/0.17/0.00	0.24/0.21/0.00	0.28/0.17/0.00	0.55/ 0.31/0.00
iGSM	ICL	4.67/56.38/1.83	6.08/55.06/3.17	5.15/58.21/2.38	6.38/64.56/1.09
	Graph SFT	78.98/54.07/34.25	77.18/ 67.48/42.97	88.00/ 67.28/51.11	86.61/60.27/47.77
	Graph-Based GRPO	99.23/61.56/43.19	97.71/65.68/50.30	98.98/60.97/50.81	90.44/64.69/48.32
MATH	ICL	4.64/22.39/1.05	11.94/31.10/3.06	6.07/34.04/1.76	14.78/41.57/0.99
	Graph SFT	34.31/39.06/15.71	35.21/44.85/18.31	43.84/44.17/21.16	54.31/52.07/30.45
	Graph-Based GRPO	82.31/44.90/38.78	87.53/47.07/41.65	88.74/47.36/42.81	69.02/54.07/37.64
ProofWriter	ICL	37.23/ 88.05/10.41	66.67/0.00/0.00	68.84/ 87.63/43.14	87.26/76.04/61.82
	Graph SFT	74.57/65.94/ 35.66	91.74/44.53/34.56	82.26/58.89/39.60	87.96/ 100.00/87.96
	Graph-Based GRPO	93.93/37.07/31.91	94.14/47.21/42.74	94.54/49.59/45.12	89.92/100.00/89.92

Table 10: Per-model faithfulness results. **(a)** Internal vs. external accuracy; a large gap indicates the emitted answer does not come from the graph. **(b)** Intervention-based metrics. Values are percentages; bold marks the best value of each metric within a model×dataset across methods. Entailment has no external answer; therefore, neither internal–external accuracy nor answer-based faithfulness metrics are applicable.

I Prompt for generating computational graph

You are an expert in computational graphs and mathematical reasoning.

Given a math word problem, generate a computational graph as a JSON array where each node represents either:

1. A constant value (type: "const")
2. An arithmetic operation (type: "op")

Rules:

1. Each constant node has: {"id": <name>, "type": "const", "value": <number>}
2. Each operation node has: {"id": <result_name>, "type": "op", "op": <operation>, "inputs": [<input_ids>]}
3. Available operations: "add", "sub", "mul", "div", "pow", "mod"
4. Operation nodes are named after their result
5. List nodes in computational order (dependencies before usage)
6. Use descriptive names for node IDs
7. Constants should represent actual numbers from the problem
8. Each operation produces a new value
9. Each operation could only use previous const and op nodes as inputs
10. After the end of the graph, write "ANSWER: <number>"

EXAMPLE:

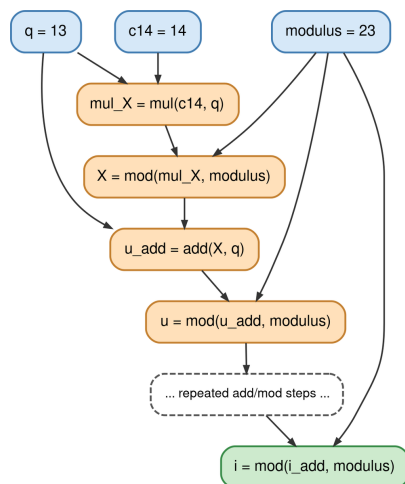
Question: Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?

```
[
  {"id": "april_clips", "type": "const", "value": 48},
  {"id": "half", "type": "const", "value": 0.5},
  {"id": "may_clips", "type": "op", "op": "mul", "inputs": [
    "april_clips", "half"]},
  {"id": "total_clips", "type": "op", "op": "add", "inputs": [
    "april_clips", "may_clips"]}
]
```

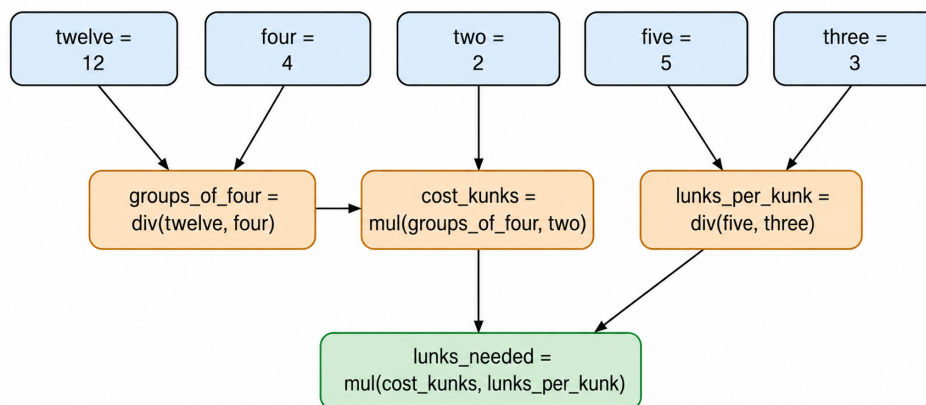
ANSWER: 72

Figure 5: Example of prompt used for generating structured computational graphs with answer.

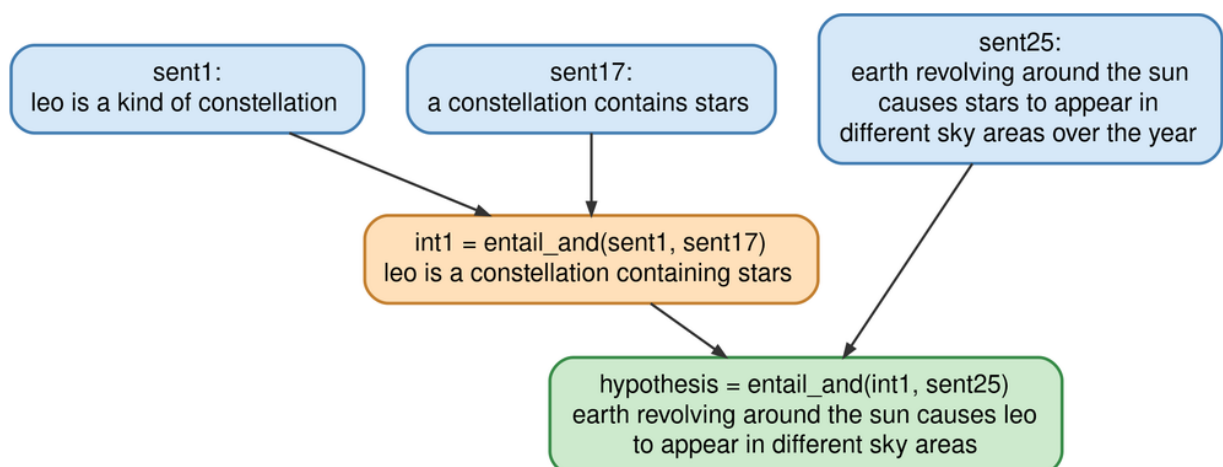
J Task examples



(a) **iGSM**. (Shortened version of the original question) Seaside / Pine Valley / Woodgrove lab counts linked by add/mod/mul relations under modular arithmetic. How many Classrooms does Seaside Community College have? **Answer**. 10.



(b) **MATH**. If 5 lunks can be traded for 3 kunks, and 2 kunks will buy 4 apples, how many lunks are needed to purchase one dozen apples? **Answer**. 10.



(c) **Entailment**. Stars are organized into patterns called constellations. One constellation is named Leo. Which statement best explains why Leo appears in different areas of the sky throughout the year? **Answer**. Earth revolves around the sun.