How Is LLM Reasoning Distracted by Irrelevant Context? An Analysis Using a Controlled Benchmark

Minglai Yang [1] **Ethan Huang** [1] **Liang Zhang** [2] **Mihai Surdeanu** [1] **William Wang** [3] **Liangming Pan** [2]

Abstract

We introduce Grade School Math with Distracting Context (GSM-DC), a synthetic benchmark to evaluate Large Language Models' (LLMs) reasoning robustness against systematically controlled irrelevant context (IC). GSM-DC constructs symbolic reasoning graphs with precise distractor injections, enabling rigorous, reproducible evaluation. Our experiments demonstrate that LLMs are significantly sensitive to IC, affecting both reasoning path selection and arithmetic accuracy. Additionally, training models with strong distractors improves performance in both in-distribution and out-of-distribution scenarios. We further propose a stepwise tree search guided by a process reward model, which notably enhances robustness in out-of-distribution conditions. The code of our dataset and experiment can be viewed at https://anonymous.4open.science/r/GSM-DC-88CC/.

1. Introduction

Recent advances in Large Language Models (LLMs) have enabled impressive reasoning capabilities across diverse tasks, particularly in solving mathematical problems (Cobbe et al., 2021b; Lewkowycz et al., 2022; Zhou et al., 2022; Yao et al., 2023c). However, these models remain vulnerable to subtle forms of distraction that impair their reasoning (Berglund et al., 2024; Huang et al., 2024; Xu et al., 2024). A well-known phenomenon in cognitive psychology—the Flanker Task (Eriksen & Eriksen, 1974)—demonstrates that human performance deteriorates when irrelevant stimuli are introduced. Drawing a parallel, Shi et al. (2023a) showed that LLMs exhibit similar susceptibility: adding even a single distractor sentence to math problems from GSM8K (Cobbe et al., 2021a) leads to a significant drop in accuracy. This suggests that, like humans, LLMs struggle to isolate relevant information in the presence of competing context.

Despite the above efforts, the mechanism behind LLMs being distracted by irrelevant context has not been systematically investigated. Shi et al. (2023a) injects one single distractor, restricts examples to short reasoning chains, and omits any form of supervised fine-tuning or out-of-distribution (OOD) evaluation. However, several key questions remain unexamined: How do varying quantities of IC affect LLM robustness? Is robust reasoning an ability to be learned by SFT, including LoRA finetuning and continued pretraining? Can we improve the model's robustness through training with IC? How much IC injected during training performs best in-distribution and OOD testing? How to automatically verify each intermediate reasoning step to ensure the model is not drawing conclusions from irrelevant context?

To address these gaps, we introduce *GSM-DC*, a synthetic benchmark designed to enable precise control over both reasoning complexity and distractor structure. Problems in *GSM-DC* are represented as symbolic dependency graphs, where nodes correspond to intermediate quantities and edges represent symbolic operations. This structure facilitates: 1) the explicit injection of irrelevant context via off-path nodes and edges without affecting correct solutions; 2) adjustment of reasoning complexity by varying graph depth and structure; and 3) automatic evaluation of model outputs by aligning predictions with the correct reasoning path.

Our dataset construction pipeline (Figure 1) involves generating symbolic dependency graphs, injecting distractors after determining the solution path, and transforming these into human-readable math word problems and solutions. We partition our dataset based on different problem complexities and distractor intensities, conduct various controlled experiments, and use automatic stepwise metrics measuring arithmetic correctness and distraction robustness. Our controlled experiments yield three main findings. First, model accuracy steadily decreases as distractor intensity

[1]Department of Computer Science, University of Arizona, Tucson, AZ, USA [2]School of Information, University of Arizona, Tucson, AZ, USA [3]Department of Computer Science, University of California, Santa Barbara, CA, USA. Correspondence to: Minglai Yang <mingly@arizona.edu>, Liangming Pan <liangmingpan@arizona.edu>.

The second AI for MATH Workshop at the 42^{nd} International Conference on Machine Learning, Vancouver, Canada.

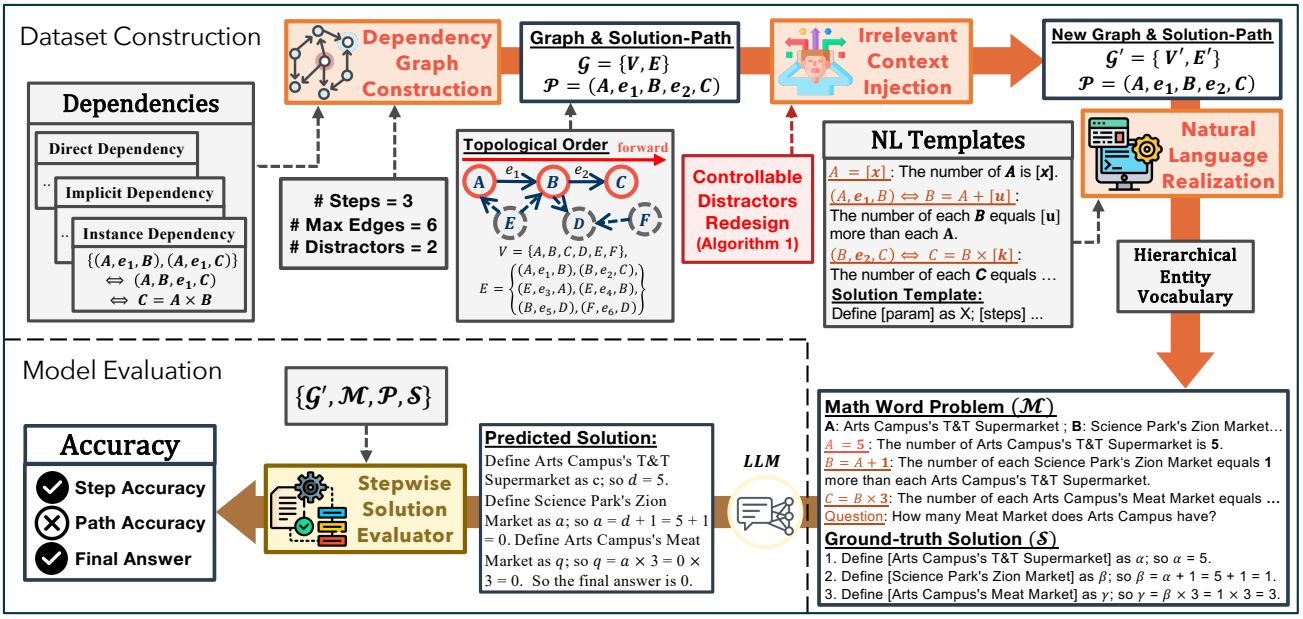

Figure 1: Overview of the *GSM-DC* framework: both generation and evaluation pipeline. The dataset construction process (orange) involves three key steps: (1) **Dependency Graph Construction** builds a symbolic DAG with a defined solution path via topological sort, (2) **Irrelevant Context Injection** adds controllable distractor nodes to increase reasoning complexity, and (3) **Natural Language Realization** converts the symbolic graph into a human-readable word problem and finds the solution following the solution path. The resulting instance is evaluated using a **Stepwise Solution Evaluator** that computes Step Accuracy, Path Accuracy, and Extraction Answer Accuracy.

rises. Second, continued pretraining substantially enhances reasoning robustness. Third, incorporating strong IC during training significantly boosts model resilience, showing superior performance across various distractor intensities in out-of-domain testing.

To improve the model's robustness against IC, we propose a stepwise beam search algorithm guided by a Process Reward Model (PRM), which scores partial reasoning paths based on their alignment with valid solution trajectories. This approach further improves robustness by up to 6.29% in out-of-domain conditions, highlighting reinforcement learning's potential to strengthen robustness against irrelevant context in model reasoning.

2. Related Work

Reasoning with Irrelevant Context LLMs often struggle to reason accurately in the presence of irrelevant context (IC). Prior work has explored this vulnerability by introducing distractors into math problems. For example, GSM-IC (Shi et al., 2023a) appends irrelevant sentences to arithmetic questions but lacks control over distractor structure or complexity. GSMIR (Jiang et al., 2024) and MPN (Song & Tavanapong, 2024) use handcrafted prompting strategies to mitigate the effects of textual noise. Anantheswaran et al. (2024) generate adversarial math problems by adding irrelevant variables, showing significant performance drops and partial robustness gains through fine-tuning. However, their hand-crafted distractors risk introducing bias and lack structural control. Other studies, such as Wu et al. (2024), show that semantically similar but irrelevant documents can impair LLM performance in retrieval settings. While these works expose LLMs' sensitivity to IC, they provide limited control over distractor properties. In contrast, *GSM-DC* injects distractors into symbolic reasoning graphs, enabling stepwise evaluation. We further show that a reward-guided beam search improves robustness beyond standard fine-tuning.

Understanding LLM Reasoning LLM reasoning has received growing attention, leading to diverse efforts to improve performance on complex tasks. Recently, synthetic benchmarks such as GSM-∞ (Zhou et al., 2025) and iGSM (Ye et al., 2024) explored LLM reasoning under long-context and complex distractors. Unlike GSM-∞ and iGSM, which respectively emphasize reasoning depth and internal mechanisms, our GSM-DC explicitly controls irrelevant distractors within symbolic DAGs to systematically quantify the effects of irrelevant context. Hao et al. (2024) introduced AutoRace and the LLM Reasoners library to standardize reasoning evaluation. CoT prompting and in-context learning have been shown to enhance logical reasoning (Bertolazzi

et al., 2024), while other work highlights limitations in handling strict deductions (Li et al., 2024). Recent methods such as ReAct (Yao et al., 2023b), Tree-of-Thoughts (Yao et al., 2023a), and self-consistency decoding (Wang et al., 2023) guide intermediate steps to improve solution quality. Beyond final-answer supervision, Process Reward Models (PRMs) (Uesato et al., 2022; Lightman et al., 2024; Zheng et al., 2024; Kumar et al., 2024; Hosseini et al., 2024) evaluate partial reasoning paths to promote more robust, interpretable, and aligned multi-step reasoning. Finally, Shi et al. (2023b) showed that few-shot abduction boosts generalization with minimal supervision.

3. The *GSM-DC* Dataset

To systematically investigate how LLMs reason under irrelevant context (IC), we require a framework that satisfies three desiderata: 1) fine-grained manipulation of IC, 2) precise control over reasoning difficulty, and 3) automatic evaluation of reasoning robustness. Existing datasets (§2) like GSM-IC are manually built and rely on free-form outputs, lacking structural constraints and making stepwise evaluation impractical without manual checks.

We propose the *Grade School Math with Distracting Context* (***GSM-DC***) benchmark—a controlled framework for systematically evaluating LLMs' reasoning under irrelevant context that meets the above criteria. Each math word problem in *GSM-DC* is represented as a directed acyclic graph (DAG), which allows us to 1) explicitly control irrelevant context by injecting distracting nodes and edges, 2) explicitly control reasoning difficulty by adjusting the graph size, and 3) automatically compute stepwise reasoning correctness by comparing model predictions to the ground-truth reasoning path. As illustrated in Figure 1, we construct the *GSM-DC* dataset in three steps:

These capabilities are enabled by our central design: a ground-truth solution path embedded within a distractor-rich but structurally constrained symbolic graph.

1) Dependency Graph Construction (§3.1): To represent a math word problem, we build a symbolic dependency graph $\mathcal{G}$ to capture the direct, implicit, and instance-level dependencies in the problem. We then identify a single correct reasoning path $\mathcal{P}$ from the graph $\mathcal{G}$ via topological sort.

2) Irrelevant Context Injection (§3.2): We turn all nodes outside the reasoning path $\mathcal{P}$ into distractors, producing an augmented graph $\mathcal{G}'$. This allows us to explicitly control the problem complexity (*e.g.*, number of reasoning steps) and the intensity of irrelevant context (*e.g.*, via the number and connectivity of distractor nodes).

3) Natural Language Realization (§3.3): We then convert the augmented graph $\mathcal{G}'$ into a human-understandable math word problem $\mathcal{M}$ by mapping each node to a real-world entity and rendering each edge into a statement. The ground-truth solution $\mathcal{S}$ is then derived from the original reasoning path $\mathcal{P}$.

As a result, each problem in the *GSM-DC* is represented as ($\mathcal{G}'$, $\mathcal{M}$, $\mathcal{P}$, $\mathcal{S}$). This structured representation enables automatic stepwise evaluation (§3.4) of LLMs' reasoning chain via the ground-truth path $\mathcal{P}$. Specifically, we introduce a stepwise solution evaluator that leverages the ground-truth path $\mathcal{P}$ to assess model outputs using three metrics: step accuracy (SAcc), path accuracy (PAcc), and final-answer correctness (EAcc). In the following, we will introduce the dataset construction pipeline in detail.

3.1. Dependency Graph Construction

Many grade-school math or logical reasoning problems involve quantities that are interrelated in various ways. These dependencies typically fall into three categories: 1) *Direct dependencies*, where one quantity is computed directly from another (*e.g.*, if R denotes the radius of a circle and T its diameter, then $T = 2 \times R$); 2) *Instance dependencies*, one entity is automatically reliant on another without explicitly stating that reliance. (*e.g.*, "Each shelf holds M books, and there are N shelves") and 3) *Implicit dependencies*, requiring aggregation or inference over multiple quantities (*e.g.*, grouping cats and dogs as animals).

To model these interrelations, we use the directed acyclic graph (DAG), denoted as $\mathcal{G}$, where each *node* denotes a quantity (*e.g.*, Bob's pens) and each *edge* represents the dependency between quantities (*e.g.*, Alice has one more pen than Bob). We name $\mathcal{G}$ as the *dependency graph*. We use DAG because the acyclicity ensures that no quantity depends on itself, allowing a valid solution path $\mathcal{P}$ to be recovered via topological sort.

This structured graph-based representation forms the foundation for controlling reasoning complexity and enables injection of irrelevant context without affecting the original solution path $\mathcal{P}$. Given inputs—reasoning steps rs, maximum edges E and distractor count m—we generate a DAG by sampling nodes and edges, then extract the solution path $\mathcal{P}$ of length rs via topological sort, and finally inject m controllable distractors (§3.2).

3.2. Irrelevant Context Injection

To create a problem with irrelevant information, we augment the dependency graph by injecting distractor nodes while preserving the original solution path. As illustrated in Figure 2 and described in Algorithm 1, we start with a clean dependency graph $\mathcal{G}$ and its solution path $\mathcal{P}$. Unused

Algorithm 1 INJECTDISTRACTORS (see Figure 2)

Input: directed acyclic graph $\mathcal{G}$, solution path P, distractor batch size m, parent–set mixing prob. q, edge-density ρ
Output: augmented graph $\mathcal{G}'$ that preserves P
$\mathcal{G}' \leftarrow \mathcal{G}$ ▷ work on a copy
$\mathcal{R} \leftarrow$ UNUSEDPARAMETERS$(\mathcal{G}', P)$
while $\mathcal{R} \neq \varnothing$ **do**
  Sample batch $\mathcal{B} \subseteq \mathcal{R}$ with $|\mathcal{B}| = m$
  for all $\chi \in \mathcal{B}$ **do**
    $\mathcal{R} \leftarrow \mathcal{R} \setminus \{\chi\}$; $n \leftarrow$ NEWNODE(χ)
    ADDNODE$(\mathcal{G}', n)$ ▷ n is a distractor
    if ISUNIQUETARGET(χ) **then**
      LABELINDEPENDENT(n) ▷ no parents
      continue
    end if
    Choose parent set $\mathcal{P} \in \{\mathcal{I}, \mathcal{C}\}$ with probability q
    ADDEDGESFORWARD$(\mathcal{G}', n, \mathcal{P}, \rho)$
    LABELCOMPUTED(n)
  end for
end while
return $(\mathcal{G}', P)$

nodes, which are not part of $\mathcal{P}$, are selected and connected to existing nodes through forward-only edges, resulting in a new graph $\mathcal{G}'$ that remains acyclic.

Problem difficulty is primarily controlled by the number of reasoning steps rs. To limit the problem complexity across instances, we constrain the input DAG $\mathcal{G}$ to have at most E edges. Given such a fixed-scale graph and its solution path $\mathcal{P}$, we inject m distractor nodes (none of which lie on $\mathcal{P}$) to produce the augmented graph $\mathcal{G}'$ (Algorithm 1). Importantly, because the total graph scale is bounded by E, longer reasoning steps occupy more of the graph structure, leaving fewer nodes and edges available for distractor injection. We vary $m \in [m_{\min}, m_{\max}]$ to define three distractor intensity levels (*e.g.*, for $rs = 2$, light uses $m \in [0,2]$, medium $m \in [3,4]$, hard $m \geq 5$). To ensure equal-sized noise levels, we compute the empirical CDF of distractor levels z_i as $\hat{F}_z(t) = \frac{1}{M}\sum_{i=1}^{M} \mathbb{I}(z_i \leq t)$ and select $m = \tau_k$ with $\hat{F}_z(\tau_k) = \frac{k}{N}$ for $k \sim \text{Uniform}\{1, \ldots, N\}$. You can see the full details are in Appendix B.

3.3. Natural Language Realization

Once the dependency graph $\mathcal{G}$ is constructed and augmented as $\mathcal{G}'$, we instantiate it into natural language. Each node is mapped to an entity (*e.g.*, "Arts Campus's T&T Supermarket") from the hierarchical entity vocabulary of the GSM8K dataset, and each edge is rendered using a templated relational statement (*e.g.*, "the number of Zion Markets is 1 more than the number of T&T Supermarkets")[1]. These templates capture the underlying dependencies while maintaining simple, readable language.

[1]We adopt the hierarchical entity vocabulary and templated relational statements introduced in (Ye et al., 2024).

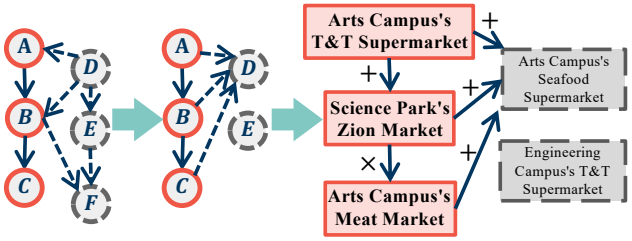

Figure 2: Distractor construction in *GSM-DC*. After generating a DAG, we retain only the original topological path used in the solution (A → B → C). Distractor nodes are constructed by adding forward edges from solution nodes to unused parameters, preserving acyclicity. Since we control which unused parameters are included and their dependencies, D is the sum of A, B and C; E becomes an independent variable; F is excluded entirely.

To form the math problem $\mathcal{M}$, we concatenate natural-language realizations of edges along the solution path, ending with a question about the final node. Distractors are rendered as unrelated sentences and shuffled with relevant content.

Alongside the natural language (NL) problem $\mathcal{M}$, we generate its corresponding NL solution $\mathcal{S}$ based on the ground-truth reasoning path $\mathcal{P}$. The solution $\mathcal{S}$ sequentially defines variables for each node along the path $\mathcal{P}$ and applies the dependencies. An example of the NL problem is given in Figure 3.

Math Word Problem

The number of each Arts Campus' T&T Supermarket equals 3. The number of each Engineering Campus' T&T Supermarket equals 4. **The number of each Science Park's Zion Market equals 1 more than each Arts Campus' T&T Supermarket.** The number of each Arts Campus' Seafood Supermarket equals the sum of each Science Park's Zion Market, Arts Campus' T&T Supermarket and each Arts Campus' Meat Market. **The number of each Arts Campus' Meat Market equals 4 times as much as each Science Park's Zion Market. How many Meat Market does Arts Campus have?**

Ground-Truth Solution

Define Arts Campus's T&T Supermarket as a; so $a = 3$. Define Science Park's Zion Market as e; so $e = a + 1 = 3 + 1 = 4$. Define Arts Campus's Meat Market as e; so r = $e \times 4 = 4 \times 4 = 1$.

Figure 3: The final reasoning problem constructed from the example in Figure 2. Irrelevant contexts are in red[2].

The generated natural language solution provides a templated Chain-of-Thought (CoT) reasoning trace, which can be parsed to automatically evaluate the stepwise reasoning correctness.

[2]We consider arithmetics mod 5 to avoid errors from computation involving large numbers. LLMs can handle arithmetic via external tools (Schick et al., 2023; Paranjape et al., 2023).

3.4. Stepwise Solution Evaluator

After constructing *GSM-DC*, we build a stepwise solution evaluator to automatically evaluate LLM-generated solutions. For each problem and predicted solution, we report three *binary* scores; for each, a value of 1 is awarded only when the stated criterion is fully satisfied.

• **Step Accuracy (SAcc):** Our symbolic parser reads the model's chain-of-thought and executes every intermediate equation in topological order. SAcc = 1 iff *all* equations are arithmetically correct *and* each step references only symbols that have already been defined. This strict all-or-nothing formulation avoids inflating performance with partially correct derivations.

• **Path Accuracy (PAcc):** To quantify *distraction robustness* we check whether the model confines its reasoning to the augmented dependency graph $\mathcal{G}'$ after injecting irrelevant context. PAcc = 1 iff (i) no extraneous nodes appear and (ii) every required dependency is present—regardless of which arithmetic operator it applies. PAcc is a relaxation of SAcc as it only requires stepwise reasoning to be correct, but not the associated values themselves.

• **Extraction Answer Accuracy (EAcc):** To capture final-answer correctness, EAcc = 1 iff the model's extracted answer exactly matches the ground truth. We report EAcc only for prompting, but our focus remains on SAcc and PAcc.

We evaluate these metrics over a large set of problems and report each as the percentage (%) of instances achieving a score of 1.

4. Experiments

4.1. Impact of Irrelevant Context

To systematically analyze how irrelevant context (IC) affects LLM reasoning, we conduct controlled experiments by injecting varying numbers of irrelevant context ($m = 1$–15) into math word problems $\mathcal{M}$ drawn from *GSM-DC* (§3). We evaluate performance across four levels of reasoning steps, denoted $rs \in \{2, 3, 4, 5\}$, and sample 100 instances per condition to ensure statistical stability.

We benchmark six models: Grok-3-Beta, GPT-4.1, GPT-4o-mini, LLaMA-3.3-70B, LLaMA-3.1-8B and LLaMA-3.2-1B. We employ a five-shot prompting strategy enhanced with a structured *Background* section (Appendix A.2) that explicitly encodes relevant dependencies to guide reasoning. Model performance is assessed using three metrics using *Stepwise Solution Evaluator*, SAcc, PAcc and EAcc, which together capture reasoning correctness, robustness to distractors, and output correctness (§3.4). This decomposition allows us to isolate the specific ways in which irrelevant context degrades model performance.

> **Result I:** *LLMs' reasoning performance degrades with increasing irrelevant context.*

As shown in Figure 4, all six models exhibit a clear degradation in reasoning accuracy as the number of irrelevant context increases. For instance, at a fixed reasoning depth of rs=5, Grok-3-Beta's step accuracy drops from 43% with one irrelevant context to just 19% under fifteen irrelevant context. GPT-4.1 exhibits an even steeper decline at the same depth, falling from 26% to 2%.

All three evaluation metrics—step accuracy (SAcc), path accuracy (PAcc), and extraction accuracy (EAcc)—exhibit similar downward trends as irrelevant context increases. Extraction accuracy (EAcc) remains relatively high, because our solution parser enforces a strict Chain-of-Thought format (§3.4) that models learn to follow through five-shot prompting. As a result, EAcc is less sensitive by distraction compared to SAcc and PAcc, which more directly assess reasoning fidelity and resistance to irrelevant information.

> **Result II:** *Irrelevant context degrades accuracy more steeply at greater reasoning depths.*

To analyze how irrelevant context (IC) interacts with reasoning complexity, we study the error rate $E(m; rs)$ as a function of distractor count m and reasoning depth rs. We find it roughly follows a power-law trend: $E(m; rs) \propto m^{\delta(rs)}$, where $\delta(rs)$ reflects a model's IC sensitivity. As shown in Figure 4, error increases with m, and the degradation steepens with deeper reasoning.

For instance, Grok-3-Beta's exponent grows from $\delta \approx 0.11$ at rs=2 to $\delta \approx 0.49$ at rs=5, indicating greater vulnerability at deeper depths. GPT-4.1 shows a similar slope but higher baseline error, suggesting that reasoning depth governs $\delta(rs)$, while model capacity sets the vertical intercept—*i.e.*, robustness under minimal distraction. These findings highlight the need to jointly consider reasoning complexity and IC sensitivity when designing robust LLMs.

4.2. Training with Different Strategies

The results so far focus on inference-time behavior: models are prompted to reason through irrelevant context (IC) without being explicitly trained on it. However, since we do not have access to the original training data of these models, it is unclear whether their observed robustness (or lack thereof) stems from genuine generalization or incidental exposure to similar patterns during pretraining. To disentangle this, we perform controlled experiments that explicitly expose models to varying degrees of IC and reasoning complexity.

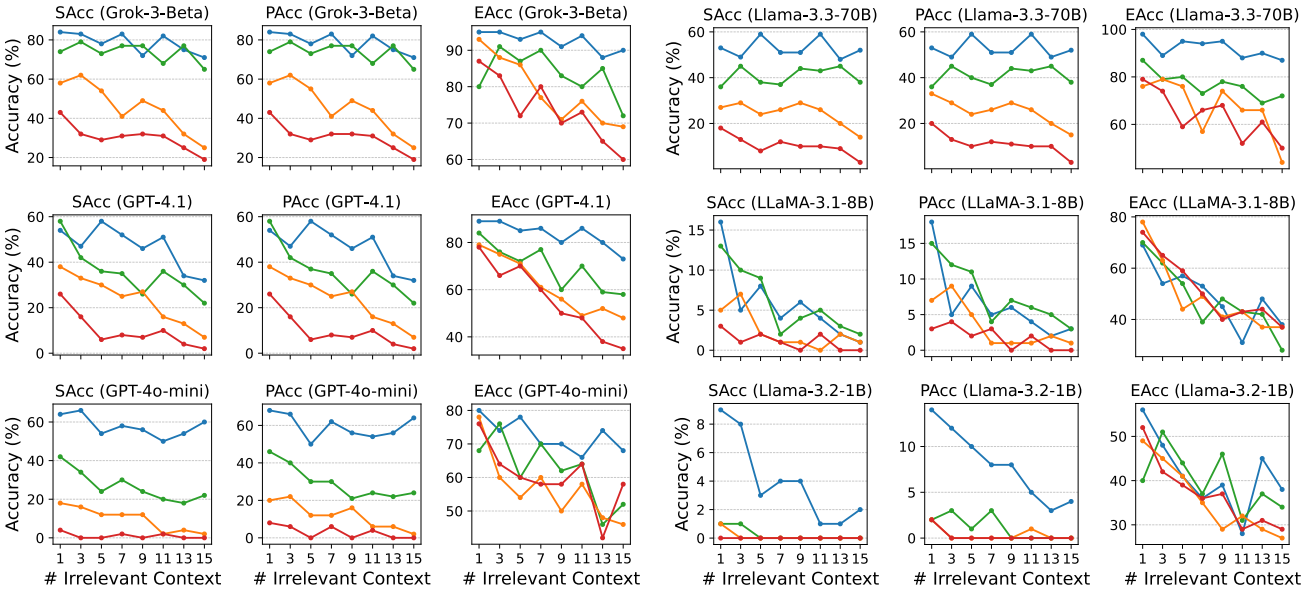

Figure 4: Step-wise accuracy under increasing irrelevant context (IC) for four models, evaluated across reasoning steps $rs \in \{2,3,4,5\}$. Each curve corresponds to a specific reasoning step: **blue** for $rs = 2$, **green** for $rs = 3$, **orange** for $rs = 4$, and **red** for $rs = 5$.

rs	Clean		Clean+IC		IC	
	SAcc	PAcc	SAcc	PAcc	SAcc	PAcc
≤ 15	35.9	41.3	70.0	71.2	**73.2**	**74.7**
16	22.0	22.7	32.0	32.0	**33.3**	**33.3**
17	21.0	21.0	23.0	23.0	**20.7**	**21.3**
18	13.0	13.0	15.7	15.7	**16.7**	**16.7**
19	13.7	13.7	13.3	13.3	**15.0**	**15.0**
20	9.0	9.0	8.3	8.3	**10.0**	**10.0**
21	7.7	7.7	8.7	8.7	**5.7**	**5.7**
22	6.0	6.0	5.3	5.3	**6.3**	**6.3**

Table 1: Comparison of SAcc and PAcc under different training regimes: Clean, Clean+IC, and IC.

First, we conduct controlled experiments on *GSM-DC* with varying reasoning steps. We first mimic the distribution in GSM-IC by training on examples with 2–7 reasoning steps, then evaluate on harder problems with up to 22 steps. As shown in Appendix C, performance drops sharply once the test depth exceeds the training horizon, suggesting that models fail to generalize if they trained with shallow reasoning samples.

To address this, we expand the training set to include examples up to rs=15, ensuring exposure to both long reasoning chains and varying levels of irrelevant context. All finetuned models in this section are trained on this broader distribution and evaluated on both *in-distribution* ($rs \leq 15$) and *out-of-distribution* ($rs > 15$) samples.

Result III: *Continued pretraining enhances robustness even without access to IC samples.*

Building on this controlled training setup, we investigate how different finetuning strategies affect reasoning robustness under irrelevant context.

Specifically, we compare continued pretraining (full finetuning) and LoRA finetuning for reasoning robustness using a 30K-sample training set, which we select based on empirical scaling trends analyzed in Appendix E. As shown in Figure 5, continued pretraining confers strong robustness even without IC supervision, substantially outperforming LoRA on clean data. With IC training, the gap narrows, but continued pretraining remains consistently more robust across reasoning depths. Based on this, we fixed continued pretraining 30K-samples for all subsequent experiments.

4.3. Control of Training Data

Result IV: *Training with Irrelevant Context Improves Robustness Most Effectively.*

As shown in Table 1, the model trained on IC consistently achieves the highest SAcc and PAcc across all *rs*. The model trained on Clean+IC data performs slightly worse, while the Non-IC model lags behind both. These results suggest that training solely on IC leads to stronger robustness.

The clean model performs worse on questions with IC, even under in-distribution (ID) settings. To better understand this limitation, we examine the gap $\Delta(\text{SAcc}, \text{PAcc})$, rep-

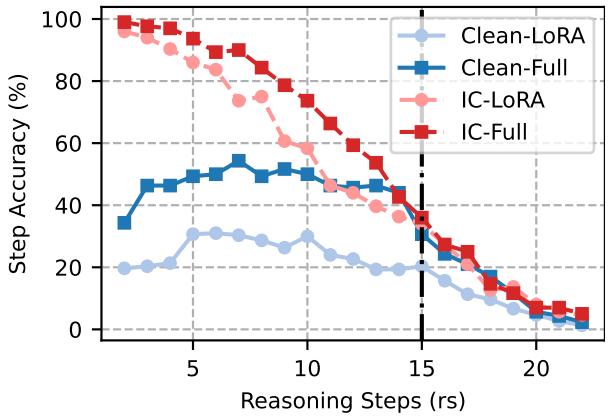

Figure 5: Step accuracy of models trained with Non-IC or IC data using LoRA or continued pretraining.

resented as the ratio between SAcc and PAcc (Figure 6). A lower ratio indicates a larger gap—arithmetic errors occurring even when the reasoning path is correct. The model trained on Clean data consistently shows a higher Δ, suggesting that IC affects not only reasoning path selection, but also arithmetic execution. These findings reveal that IC broadly disrupts reasoning, and that training with IC-injected examples leads to more robust models.

Result V: *Training with challenging irrelevant context leads to the strongest robustness and generalization across all pretraining settings.*

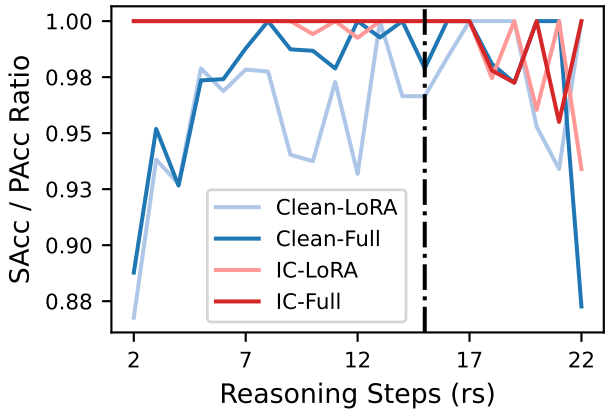

Figure 6: Step accuracy (%) for models trained with *Clean* or *IC* using LoRA or continued pretraining.

Having established that exposure to irrelevant context during training improves robustness, we now investigate whether the intensity of such context further influences generalization. In particular, we test whether training on harder, more distracting IC leads to greater robustness on out-of-distribution (OOD) reasoning problems. Based on the method described in §3, we construct two main versions of the *GSM-DC* benchmark for evaluation:

***GSM-DC*-Clean:** For each reasoning step $rs \in [2, 22] \cap \mathbb{Z}$, we sample 300 clean dependency graphs without injecting any IC. Each graph contains a unique solution path $\mathcal{P}$ and no distractor nodes. This clean subset comprises 6,300 math problems.

***GSM-DC*-with-IC:** To study robustness under IC, we generate IC variants of the clean graphs by injecting distractors following the procedure (§3.2). For each reasoning step, we sample 100 graphs under each of three IC intensity levels: LIGHT-IC, MEDIUM-IC, and HARD-IC, while keeping the reasoning path $\mathcal{P}$ fixed. Each subset thus contains 2,100 problems (100 per step), resulting in a total of 6,300 problems across all IC levels.

To evaluate how IC difficulty affects training, we compare five regimes: CLEAN, LIGHT / MEDIUM / HARD-IC, and MIX-IC. As shown in Table 2 and Table 3, HARD-IC yields the best SAcc across all in-distribution and OOD settings, regardless of IC presence or difficulty.

Training IC Level	Testing w/ IC (SAcc)			Testing w/o IC (SAcc)		
	ID	OOD	All	ID	OOD	All
CLEAN	35.91	13.19	32.36	81.95	17.05	60.32
LIGHT-IC	64.79	6.90	46.57	67.33	7.09	46.56
MEDIUM-IC	65.79	7.23	47.44	69.39	9.95	50.38
HARD-IC	**77.95**	**18.57**	**59.48**	**82.30**	**19.86**	**61.21**
MIX-IC	73.23	15.33	57.86	78.09	15.62	57.38

Table 2: Step Accuracy (%) under different training IC difficulties, evaluated across test IC conditions.

These findings indicate that exposure to adding challenging distractors (HARD-IC) is the most effective training strategy for enhancing model robustness and generalization performance. Intriguingly, MIX-IC, despite incorporating distractor diversity, consistently underperformed HARD-IC, suggesting that distractor difficulty, rather than variety, is the primary driver of improvement. The advantage of HARD-IC over NON-IC, particularly under test-time IC conditions, further reinforces the utility of IC augmentation, specifically with high-difficulty examples, for fostering robust reasoning.

5. Improving Model Robustness Against Irrelevant Context

The previous section (§4) demonstrate that LLMs are highly sensitive to irrelevant context (IC), and that continued pretraining with challenging IC-injected examples alone can substantially improve robustness. However, even with the strongest continued pretraining configurations (*e.g.*, HARD-IC), model performance still degrades significantly on out-of-distribution (OOD) reasoning steps. This raises the question of how robustness can be further improved at test time.

Training IC Level	ID Test SAcc			OOD Test SAcc		
	Light	Medium	Hard	Light	Medium	Hard
LIGHT-IC	67.21	66.57	60.57	8.14	7.29	5.28
MEDIUM-IC	68.14	66.07	63.14	8.71	8.43	4.57
HARD-IC	**78.36**	**79.21**	**76.28**	**22.7**	**18.43**	**14.57**
MIX-IC	74.71	75.07	69.93	17.7	16.57	11.28

Table 3: Step Accuracy (%) per test IC difficulty. All models are trained with a specific IC difficulty.

5.1. Tree of Thoughts

Our Tree of Thoughts (ToT) algorithm addresses complex reasoning problems by combining tree search with the step-by-step inference capabilities of large language models (LLMs). As illustrated in Figure 7, ToT not only uses an LLM to propose candidate reasoning steps, but also integrates a Process Reward Model (PRM) to evaluate and guide the search process. Given a partial reasoning path $h_{1:t}$, the PRM assigns a reward $R(h_{1:t})$ indicating the quality of reasoning up to step t. Leveraging a synthetic dataset, we systematically inject irrelevant context (IC) and arithmetic errors into selected reasoning paths. These negative examples are used to train the PRM to distinguish valid reasoning trajectories from those corrupted by irrelevant context (IC) and wrong arithmetic calculations enabling the model to prioritize more accurate and robust solutions during search.

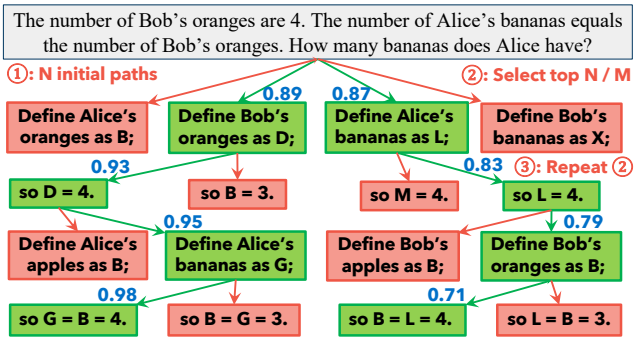

Figure 7: An overview of the ToT algorithm with $N = 4$ and $M = 2$. Green nodes indicate those that were scored highly by the PRM and thus expanded in subsequent iterations, while red nodes were not selected as candidate nodes for the next step. After the algorithm terminates, the leftmost node is scored the highest and thus that reasoning path is chosen as the final answer.

5.2. Experiments

In our experiments, the problem set that each model was tested on consisted of 300 total questions per OP level with 100 questions of each intensity level (light, medium, and hard). This was done to ensure that all types of problems of varying degrees of difficulty were fairly represented within each experimental setting. In settings without a PRM, we allowed the LLM to generate a response until it generated the <EOS> token. We would then take this generated response and pass it into our parser to determine whether or not it was correct. In settings with a PRM, we generated responses in a step by step manner by using ";" and "." as our intermediary stop tokens. Each intermediary step would be scored by the PRM and only the top N/M responses would be selected as candidates in the next step to be explored further. This process was repeated until the LLM generated the <EOS> token, signaling that the response was complete. Similar to the non-PRM case, this final response would then be passed into the parser to determine its correctness.

Training IC Level	ID SAcc			OOD SAcc		
	w/o PRM	w/ PRM	Δ	w/o PRM	w/ PRM	Δ
LIGHT-IC	64.79	**66.10**	+1.31	6.90	**9.59**	+2.69
MEDIUM-IC	65.79	**70.05**	+4.26	7.23	**13.52**	+6.29
HARD-IC	77.95	**79.48**	+1.53	18.57	**24.17**	+5.60
MIX-IC	73.23	**75.81**	+2.58	15.33	**19.06**	+3.73
CLEAN	35.91	**36.38**	+0.47	13.19	**15.76**	+2.57

Table 4: The Step and Path Accuracies of the different model types without and with PRM.

The results suggest that using a PRM preserves model performance in ID tasks, while also allowing the model to generalize its responses to OOD tasks. Table 4 presents our experimental results for each setting and their corresponding accuracies. As can be seen from the table, out of all models trained with varying degrees of IC, the model trained with hard IC performs the greatest, and supplementing it with a PRM significantly improves its accuracy.

6. Conclusion

We present ***GSM-DC***, a controlled benchmark designed to rigorously evaluate and improve the robustness of LLM reasoning in the presence of systematically injected irrelevant context (IC). By framing math problems as symbolic directed acyclic graphs, *GSM-DC* enables precise control over reasoning complexity and distractor structure, along with automatic stepwise evaluator. Our experiments reveal that: 1) LLM accuracy degrades as distractor count increases, with the error roughly following a power-law trend whose exponent grows with reasoning depth; 2) IC affects not only reasoning path selection, but also arithmetic execution; 3) Training with challenging IC, combined with continued pretraining, yields the strongest robustness across both ID and OOD settings, outperforming LoRA finetuning under clean and noisy conditions. Finally, we show that reasoning robustness can be further improved at test time using beam

search with PRM, which boosts OOD step accuracy by up to 6.29%. Together, these findings position *GSM-DC* as both a diagnostic tool for analyzing IC sensitivity and a foundation for developing robust training and inference strategies for language models reasoning.

Limitations

GSM-DC provides a controlled environment for probing LLM reasoning, combining symbolic DAGs with natural-language templates inspired by datasets like iGSM (Ye et al., 2024). To enhance linguistic diversity and realism, we designed a hierarchical vocabulary system derived from GSM8K (Cobbe et al., 2021a) and constructed templated prompts with varied surface forms. While this approach balances control and naturalness, the use of templates still limits full linguistic expressiveness. To address this, we plan to expand the benchmark with more diverse natural-language realizations sampled from real corpora and support more flexible arithmetic reasoning. The current reasoning depth is capped at 22 operations; we are generating new tiers with 30+ steps to explore long-horizon compositionality. While we benchmark six models—Grok-3-Beta, GPT-4.1, GPT-4o-mini, LLaMA-3.3-70B, LLaMA-3.1-8B, and LLaMA-3.2-1B—all training experiments are conducted solely on LLaMA-3.2-1B using a 30K-sample dataset (see Appendix E) due to computational constraints. Future work will scale to larger models to test robustness across capacities. To generalize our Process Reward Model and Tree-of-Thoughts framework, we will retrain the PRM on external reasoning datasets (*e.g.*, ProofWriter, StrategyQA) and benchmark adaptive beam heuristics. Finally, we aim to include faithfulness and bias diagnostics—such as explanation consistency and demographic sensitivity—to ensure that robustness gains translate into safe and trustworthy reasoning.

Impact Statement

This research uses only synthetic data and does not involve human subjects or sensitive information. All models and experiments comply with the licenses of publicly available tools. We support responsible AI research and have prioritized transparency and reproducibility throughout this work.

References

Allen-Zhu, Z. and Li, Y. Physics of language models: Part 1, learning hierarchical language structures, 2024a. URL https://arxiv.org/abs/2305.13673.

Allen-Zhu, Z. and Li, Y. Physics of language models: Part 3.2, knowledge manipulation, 2024b. URL https://arxiv.org/abs/2309.14402.

Allen-Zhu, Z. and Li, Y. Physics of language models: Part 3.3, knowledge capacity scaling laws, 2024c. URL https://arxiv.org/abs/2404.05405.

Anantheswaran, U., Gupta, H., Scaria, K., Verma, S., Baral, C., and Mishra, S. Cutting through the noise: Boosting

llm performance on math word problems. *arXiv preprint arXiv:2406.15444*, 2024.

Berglund, L., Tong, M., Kaufmann, M., Balesni, M., Stickland, A. C., Korbak, T., and Evans, O. The reversal curse: Llms trained on "a is b" fail to learn "b is a", 2024. URL https://arxiv.org/abs/2309.12288.

Bertolazzi, L., Gatt, A., and Bernardi, R. A systematic analysis of large language models as soft reasoners: The case of syllogistic inferences. In Al-Onaizan, Y., Bansal, M., and Chen, Y. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 13882–13905. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.emnlp-main.769.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021a.

Cobbe, K., Kosaraju, V., Bavarian, M., Hilton, J., Nakano, R., Schulman, J., and Irving, G. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021b.

Eriksen, B. A. and Eriksen, C. W. Effects of noise letters upon the identification of a target letter in a nonsearch task. *Perception & Psychophysics*, 16:143–149, 1974. URL https://api.semanticscholar.org/CorpusID:12012872.

Hao, S., Gu, Y., Luo, H., Liu, T., Shao, X., Wang, X., Xie, S., Ma, H., Samavedhi, A., Gao, Q., Wang, Z., and Hu, Z. LLM reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models. *CoRR*, abs/2404.05221, 2024. doi: 10.48550/ARXIV.2404.05221. URL https://doi.org/10.48550/arXiv.2404.05221.

Hosseini, A., Yuan, X., Malkin, N., Courville, A. C., Sordoni, A., and Agarwal, R. V-star: Training verifiers for self-taught reasoners. *CoRR*, abs/2402.06457, 2024. doi: 10.48550/ARXIV.2402.06457. URL https://doi.org/10.48550/arXiv.2402.06457.

Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. Large language models cannot self-correct reasoning yet, 2024. URL https://arxiv.org/abs/2310.01798.

Jiang, M., Huang, T., Guo, B., Lu, Y., and Zhang, F. Enhancing robustness in large language models: Prompting for mitigating the impact of irrelevant information. *CoRR*, abs/2408.10615, 2024. doi: 10.48550/ARXIV.2408.10615. URL https://doi.org/10.48550/arXiv.2408.10615.

Kumar, A., Zhuang, V., Agarwal, R., Su, Y., Co-Reyes, J. D., Singh, A., Baumli, K., Iqbal, S., Bishop, C., Roelofs, R., Zhang, L. M., McKinney, K., Shrivastava, D., Paduraru, C., Tucker, G., Precup, D., Behbahani, F. M. P., and Faust, A. Training language models to self-correct via reinforcement learning. *CoRR*, abs/2409.12917, 2024. doi: 10.48550/ARXIV.2409.12917. URL https://doi.org/10.48550/arXiv.2409.12917.

Lewkowycz, A., Zelikman, E., Tao, R., Hashimoto, T., Schulman, J., Chen, X., Zitnick, C. L., Manning, C. D., Saunders, D., Santoro, A., et al. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

Li, Z., Cao, Y., Xu, X., Jiang, J., Liu, X., Teo, Y. S., Lin, S., and Liu, Y. Llms for relational reasoning: How far are we? In *LLM4CODE@ICSE*, pp. 119–126, 2024. doi: 10.1145/3643795.3648387. URL https://doi.org/10.1145/3643795.3648387.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Paranjape, B., Lundberg, S., Singh, S., Hajishirzi, H., Zettlemoyer, L., and Ribeiro, M. T. Art: Automatic multi-step reasoning and tool-use for large language models, 2023. URL https://arxiv.org/abs/2303.09014.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools, 2023. URL https://arxiv.org/abs/2302.04761.

Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E. H., Schärli, N., and Zhou, D. Large language models can be easily distracted by irrelevant context. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31210–31227. PMLR, 2023a. URL https://proceedings.mlr.press/v202/shi23a.html.

Shi, X., Xue, S., Wang, K., Zhou, F., Zhang, J. Y., Zhou, J., Tan, C., and Mei, H. Language models

can improve event prediction by few-shot abductive reasoning. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023b. URL http://papers.nips.cc/paper_files/paper/2023/hash/5e5fd18f863cbe6d8ae392a93fd271c9-Abstract-Conference.html.

Song, S. H. and Tavanapong, W. How much do prompting methods help llms on quantitative reasoning with irrelevant information? In Serra, E. and Spezzano, F. (eds.), *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024, Boise, ID, USA, October 21-25, 2024*, pp. 2128–2137. ACM, 2024. doi: 10.1145/3627673.3679840. URL https://doi.org/10.1145/3627673.3679840.

Uesato, J., Kushman, N., Kumar, R., Song, H. F., Siegel, N. Y., Wang, L., Creswell, A., Irving, G., and Higgins, I. Solving math word problems with process- and outcome-based feedback. *CoRR*, abs/2211.14275, 2022. doi: 10.48550/ARXIV.2211.14275. URL https://doi.org/10.48550/arXiv.2211.14275.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Wu, S., Xie, J., Chen, J., Zhu, T., Zhang, K., and Xiao, Y. How easily do irrelevant inputs skew the responses of large language models? *CoRR*, abs/2404.03302, 2024. doi: 10.48550/ARXIV.2404.03302. URL https://doi.org/10.48550/arXiv.2404.03302.

Xu, W., Zhu, G., Zhao, X., Pan, L., Li, L., and Wang, W. Y. Pride and prejudice: Llm amplifies self-bias in self-refinement, 2024. URL https://arxiv.org/abs/2402.11436.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023a. URL http://papers.nips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.

Yao, S., Zhao, J., Yu, D., Zhao, I., Yu, Y., Gao, E., Bosma, M., Wang, Y., and Zhou, D. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023c.

Ye, T., Xu, Z., Li, Y., and Allen-Zhu, Z. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. *CoRR*, abs/2407.20311, 2024. doi: 10.48550/ARXIV.2407.20311. URL https://doi.org/10.48550/arXiv.2407.20311.

Zheng, C., Zhang, Z., Zhang, B., Lin, R., Lu, K., Yu, B., Liu, D., Zhou, J., and Lin, J. Processbench: Identifying process errors in mathematical reasoning. *CoRR*, abs/2412.06559, 2024. doi: 10.48550/ARXIV.2412.06559. URL https://doi.org/10.48550/arXiv.2412.06559.

Zhou, X., Schärli, N., Hou, L., Wei, J., Mishra, S., Huang, X., Le, Q., Chung, H. W., Pham, H., Zoph, B., et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

Zhou, Y., Liu, H., Chen, Z., Tian, Y., and Chen, B. Gsm-infinite: How do your llms behave over infinitely increasing context length and reasoning complexity? *CoRR*, abs/2502.05252, 2025. doi: 10.48550/ARXIV.2502.05252. URL https://doi.org/10.48550/arXiv.2502.05252.

Content of Appendix

A. Dataset Samples

A.1. Training Dataset with Different IC for Finetuned Model

For models that have been finetuned on mathematical reasoning tasks, we provide the question directly, omitting any system or instruction prompt.

Light-IC Sample (Operations = 2)

► Input:
The number of each Arts Campus's T&T Supermarket equals 3. The number of each Science Park's Zion Market equals 1 more than each Arts Campus's T&T Supermarket. The number of each Engineering Campus's Zion Market equals each Engineering Campus's T&T Supermarket. **How many Zion Market does Science Park have?**

► Output:
Define Arts Campus's T&T Supermarket as e; so e = 3. Define Science Park's Zion Market as w; so w = e + 1 = 3 + 1 = 4.

Medium-IC Sample (Operations = 2)

► Input:
The number of each Arts Campus's T&T Supermarket equals 3. The number of each Arts Campus's La Michoacana Meat Market equals 4. The number of each Preparatory School District's La Michoacana Meat Market equals 3 more than the difference of each Science Park's T&T Supermarket and each Science Park's La Michoacana Meat Market. **The number of each Science Park's Zion Market equals 1 more than each Arts Campus's T&T Supermarket.** The number of each Engineering Campus's Zion Market equals each Engineering Campus's T&T Supermarket. **How many Zion Market does Science Park have?**

► Output:
Define Arts Campus's T&T Supermarket as e; so e = 3. Define Science Park's Zion Market as w; so w = e + 1 = 3 + 1 = 4.

Hard-IC Sample (Operations = 2)

► Input:
The number of each Arts Campus's La Michoacana Meat Market equals 4. **The number of each Arts Campus's T&T Supermarket equals 3.** The number of each Arts Campus's Seafood City Supermarket equals 2 more than each Science Park's Zion Market. The number of each Preparatory School District's Zion Market equals each Engineering Campus's Seafood City Supermarket. The number of each Science Park's Seafood City Supermarket equals the sum of each Science Park's La Michoacana Meat Market and each Science Park's T&T Supermarket. The number of each Preparatory School District's Seafood City Supermarket equals 4 more than the sum of each Science Park's Zion Market, each Arts Campus's T&T Supermarket and each Arts Campus's Seafood City Supermarket. The number of each Arts Campus's Zion Market equals the sum of each Science Park's T&T Supermarket, each Arts Campus's T&T Supermarket and each Engineering Campus's La Michoacana Meat Market. The number of each Preparatory School District's T&T Supermarket equals 4 more than each Engineering Campus's Seafood City Supermarket. The number of each Science Park's T&T Supermarket equals 4. The number of each Engineering Campus's La Michoacana Meat Market equals 0. The number of each Engineering Campus's T&T Supermarket equals 1 times as much as the difference of each Engineering Campus's La Michoacana Meat Market and each Preparatory School District's Seafood City Supermarket. The number of each Engineering Campus's Seafood City Supermarket equals 2 times as much as the sum of each Science Park's Seafood City Supermarket, each Science Park's La Michoacana Meat Market

and each Science Park's T&T Supermarket. The number of each Science Park's La Michoacana Meat Market equals 3 times as much as each Science Park's T&T Supermarket. The number of each Preparatory School District's La Michoacana Meat Market equals 3 more than the difference of each Science Park's T&T Supermarket and each Science Park's La Michoacana Meat Market. **The number of each Science Park's Zion Market equals 1 more than each Arts Campus's T&T Supermarket.** The number of each Engineering Campus's Zion Market equals each Engineering Campus's T&T Supermarket. **How many Zion Market does Science Park have?**

► **Output:**
Define Arts Campus's T&T Supermarket as e; so e = 3. Define Science Park's Zion Market as w; so w = e + 1 = 3 + 1 = 4.

Non-IC Sample (Operations = 2)

► **Input:**
The number of each Arts Campus's T&T Supermarket equals 3. The number of each Science Park's Zion Market equals 1 more than each Arts Campus's T&T Supermarket. How many Zion Market does Science Park have?

► **Output:**
Define Arts Campus's T&T Supermarket as e; so e = 3. Define Science Park's Zion Market as w; so w = e + 1 = 3 + 1 = 4.

A.2. Testing on Closed-Sourced Model

To evaluate closed-source models, we use GPT-4o-mini to test across all operations. Additionally, we included a Background from the underlying graph structure to explicitly tell the model entity relationships, helping the model construct the correct reasoning context. Since the model struggles to learn modular operations, we also embed five-shot prompting.

5-shots Testing Sample(Operations = 2)

★ **System:**
You're an expert at solving elementary math problems involving addition, subtraction, and multiplication. You solve all the problems in a uniform format. All calculations are done modulo 5. For example, 3 + 2 equals 0, 1 + 1 equals 2, 4 + 2 + 4 equals 0, 3 * 2 equals 1, and 3 * 1 equals 3. When providing your solution, please end with 'The final answer is «x».' where x is your final answer, an integer between 0 and 4. You must solve all the problems using the same solution format. Our scenarios involve up to four categories of objects:schools, classrooms, backpacks and stationeries. Each school may contain classrooms, each classroom may contain backpacks, and each backpack may contain stationeries. We can specify quantities, such as "the number of dance studios at each Lakeshore High."

Assume that every entity with the same name has an identical configuration; for example, each Lakeshore High contains the same number of dance studios. Another guiding principle is that what is not mentioned does not exist: when we refer to classrooms at Lakeshore High, we are only discussing the classrooms explicitly mentioned in our scenario. Furthermore, if Lakeshore High is not even mentioned, any classroom within it is automatically considered to be non-existent (i.e. 0).

► **User:** ...

► **Assistant:** ...

► **User:** ...
► **Assistant:** ...

► **User:** ...
► **Assistant:** ...

► **User:** ...
► **Assistant:** ...

► **User:** ...
► **Assistant:** ...

► **User:**
Background:
`There are 4 types of Zoo: Jurong Bird Park, Flamingo Gardens, Tracy Aviary, and Avery Island. There are 4 types of Enclosure: Ladybug Loft, Dragonfly Delta, Snail Shellter, and Beetle Bungalow. There are 2 types of Animal: Fire Salamander, and Newt. There are 3 types of Bone: Tertials, Secondary Feathers, and Metacarpals. Each Ladybug Loft's Fire Salamander can have Ladybug Loft's Animal. Each Tracy Aviary's Snail Shellter can have Snail Shellter's Newt and Ladybug Loft's Fire Salamander. Each Snail Shellter's Newt can have Ladybug Loft's Fire Salamander. Each Jurong Bird Park's Zoo can have Tracy Aviary's Snail Shellter, Snail Shellter's Newt, and Ladybug Loft's Fire Salamander.`

The problem description is:
`The number of each Snail Shellter's Newt equals 4 more than each Tracy Aviary's Snail Shellter. The number of each Ladybug Loft's Fire Salamander equals 1 times as much as the difference of each Snail Shellter's Newt and each Tracy Aviary's Snail Shellter. The number of each Tracy Aviary's Snail Shellter equals 4. How many Animal does Ladybug Loft have?`

► **GPT-4o-mini Predicted Solution: (Incorrect)**
Define Tracy Aviary's Snail Shellter as T; so T = 4.
Define Snail Shellter's Newt as N; so N = T + 4 = 4 + 4 = 3.
Define Ladybug Loft's Fire Salamander as F; so F = N - T = 3 - 4 = 0. **[CALCULATION ERROR]**
Define Ladybug Loft's Animal as A; so A = F = 0.

► **Groud-Truth Solution: (Correct)**
Define Tracy Aviary's Snail Shellter as o; so o = 4.
Define Snail Shellter's Newt as S; so S = 4 + o = 4 + 4 = 3.
Define Ladybug Loft's Fire Salamander as s; m = S - o = 3 - 4 = 4; so s = 1 * m = 1 * 4 = 4.
Define Ladybug Loft's Animal as H; so H = s = 4.

- **Step Accuracy:** False
- **Path Accuracy:** True
- **Final Answer Accuracy:** False

◄ **Failure Reason:**
The model correctly selects every relevant entity and follows the intended dependency chain—first computing the Newt count N from the Snail Shellter count T, then deriving the Fire Salamander count F from N and T, and finally mapping F to the total Animals—showing no influence from irrelevant context (Path Accuracy = True). Nonetheless, it commits a modular–arithmetic error: it evaluates $F = N - T = 3 - 4$ as 0 instead of the correct value 4 under modulo 5.(Step Accuracy = False, Final Answer Accuracy = False).

B. Quantifying Irrelevant Information

To empirically study the impact of irrelevant information, we control the number of extraneous nodes and edges injected into each example (see Table 5). These irrelevant parameters are randomly sampled from unused entities in the underlying graph, ensuring they do not alter the correct reasoning path. We incrementally adjust the number of injected nodes based on both model performance and problem difficulty. Notably, when the number of irrelevant nodes becomes large, model performance drops significantly. To avoid saturating the model's capacity and distorting evaluation, we refrain from injecting more irrelevant information beyond this point.

Operation	Irrelevant Parameters		
	Light	Medium	Hard
$op = 2$	0–2	3–4	5-
$op = 3$	0–1	2–4	5–
$op = 4$	0–1	2–3	4–
$op = 5$	0–1	2–3	4–
$op = 6$	0–1	2–3	4–
$op = 7$	0–1	2–3	4–
$op = 8$	0–1	2–3	4–
$op = 9$	0–1	2–2	3–
$op = 10$	0–1	2–2	3–
$op = 11$	0–0	1–2	3–
$op = 12$	0–0	1–2	3–
$op = 13$	0–0	1–2	3–
$op = 14$	0–0	1–2	3–
$op = 15$	0–0	1–2	3–
$op = 16$	0–0	1–1	2–
$op = 17$	0–0	1–1	2–
$op = 18$	0–0	1–1	2–
$op = 19$	0–0	1–1	2–
$op = 20$	0–0	1–1	2–
$op = 21$	0–0	1–1	2–

Table 5: Quantile distribution of extraneous nodes across different operations.

C. Operation-Range Bias in GSM-IC

We found that models trained on problems containing only a small number of arithmetic operations tend to overfit short reasoning templates and fail to extrapolate to longer chains of computation. To make this limitation explicit, we **adopt exactly the same operation-count distribution as GSM-IC** for all in-distribution (ID) training examples (OP $=$ 2–15). Generalisation is then probed with a held-out out-of-distribution (OOD) slice comprising problems that require sixteen to twenty-two operations. Figure 8 plots test accuracy against operation count: performance remains high within the ID range but deteriorates rapidly once the task exceeds the training horizon, underscoring the necessity of our two-tier protocol for a fair assessment of compositional reasoning.

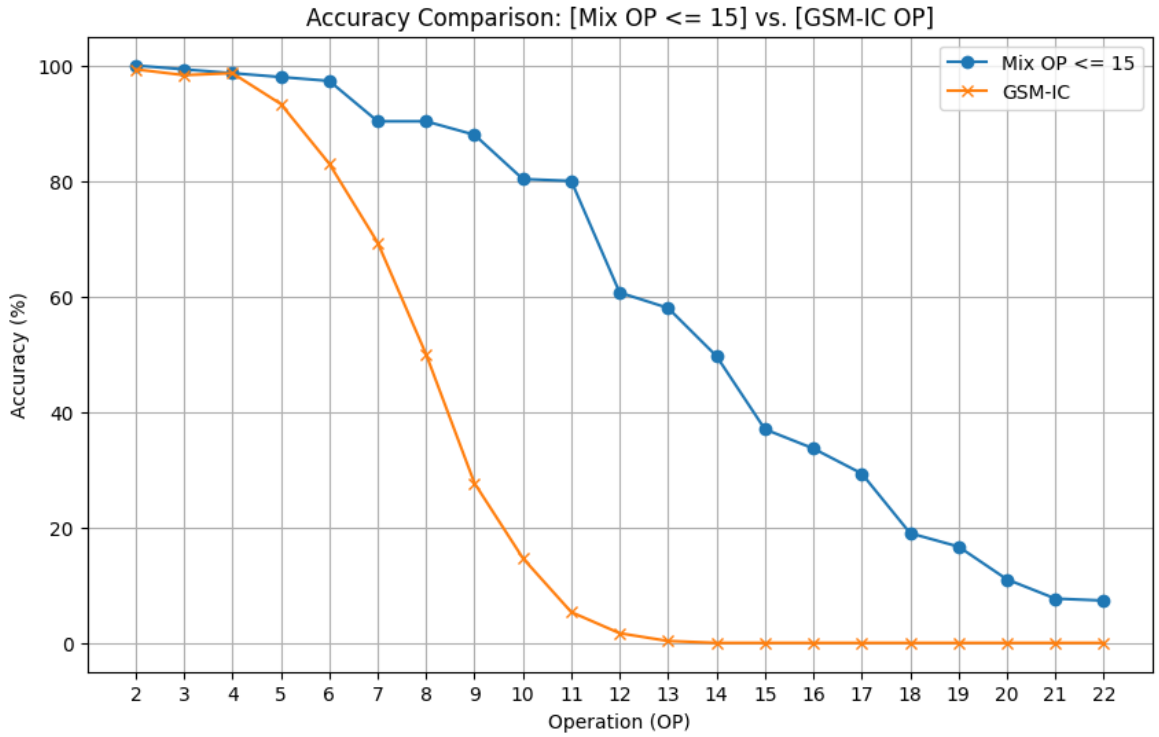

Figure 8: Test accuracy of a model trained on the MIX ≤ 15 set (operation distribution identical to GSM-IC) when evaluated on OP $=$ 2–22. Accuracy drops sharply once the required operation count exceeds the training ceiling.

D. Process Reward Model

The Process Reward Model (PRM) we used in our experiments was trained using responses from an already fine-tuned model on problems of varying operation levels. Additionally, we utilized a parser that would identify exactly at which step of the LLM's reasoning chain the first mistake was made. We segmented the model's responses by using "." and ";" as our stop tokens, and labeled each segment depending on whether it is correct or not as illustrated below. Wherever the parser identified an error, that step and all subsequent steps would receive a negative label [−], while all steps prior received a positive label [+].

PRM Example with Correct Steps

► **Problem:**
The number of each Coniferous Forest's Hippopotamus equals 2 more than the difference of each Hippopotamus's Aortic Valve and each Hippopotamus's Organs. The number of each Tropical Dry Forest's Rhinoceros equals 2 more than the sum of each Coniferous Forest's Organs, each Rhinoceros's Organs and each Hippopotamus's Cardiac Muscle. The number of each Hippopotamus's Aortic Valve equals 1. The number of each Hippopotamus's Cardiac Muscle equals 3 times as much as each Hippopotamus's Aortic Valve. How many Organs does Tropical Dry Forest have?

► **Responses + Labels:**
"Define Hippopotamus's Aortic Valve as g;" [+]
"so g = 1." [+]
"Define Hippopotamus's Cardiac Muscle as l;" [+]
"so l = 3 * g = 3 * 1 = 3." [+]
"Define Hippopotamus's Organs as G;" [+]
"so G = g + l = 1 + 3 = 4." [+]
"Define Coniferous Forest's Hippopotamus as y;" [+]
"w = g - G = 1 - 4 = 2;', 'so y = 2 + w = 2 + 2 = 4." [+]
"Define Coniferous Forest's Organs as L;" [+]
"so L = y * G = 4 * 4 = 1." [+]
"Define Rhinoceros's Organs as u;" [+]
"so u = 0." [+]
"Define Tropical Dry Forest's Rhinoceros as R;" [+]
"s = L + u = 1 + 0 = 1;" [+]
"C = s + l = 1 + 3 = 4;" [+]
"so R = 2 + C = 2 + 4 = 1." [+]
"Define Tropical Dry Forest's Organs as V;" [+]
"so V = R * u = 1 * 0 = 0." [+]

The use of a synthetic dataset offers two key advantages: it simplifies the curation of training data for our Process Reward Model (PRM) and enables precise control over injected mistakes, allowing us to label incorrect reasoning steps explicitly. This was made possible by a solution parser capable of not only verifying the final answer but also pinpointing the exact step where an error occurred. As a result, we were able to construct high-quality, fine-grained supervision signals to effectively train the PRM.

Furthermore, we are able to purposely inject IC into a given problem due to the synthetic nature of our dataset. For our experiments, after having constructed the directed graph of the correct problem, we add IC by generating extraneous details and directly including them within the given problem statement at various steps. We then pass this new problem into the LLM to get its response. Afterwards, we pass the LLM's response into our parser and it can identify the presence of and type of error that was made. Examples of errors include, but are not limited to, arithmetic errors, definitions of irrelevant/non-existent symbols, or duplicate symbols.

PRM Example with Wrong Steps

► **Problem:**
The number of each Nasal Cavity's Pericytes equals the difference of each Nasal Cavity's Arrector Pili Muscle Cells and each Vocal Cords's Arrector Pili Muscle Cells. The number of each Nasal Cavity's Gastrointestinal Smooth Muscle Cells equals 3 times as much as each Vocal Cords's Arrector Pili Muscle Cells. The number of each Respiratory Mucosa's Pericytes equals 4 times as much as the sum of each Nasal Cavity's Gastrointestinal Smooth Muscle Cells and each Vocal Cords's Arrector Pili Muscle Cells. The number of each Vocal Cords's Pericytes equals 3. The number of each Respiratory Mucosa's Arrector Pili Muscle Cells equals 2. The number of each Nasal Cavity's Arrector Pili Muscle Cells equals 3 more than the difference of each

Nasal Cavity's Gastrointestinal Smooth Muscle Cells and each Respiratory Mucosa's Pericytes. The number of each Vocal Cords's Arrector Pili Muscle Cells equals 3 times as much as each Vocal Cords's Gastrointestinal Smooth Muscle Cells. The number of each Respiratory Mucosa's Gastrointestinal Smooth Muscle Cells equals each Nasal Cavity's Arrector Pili Muscle Cells. The number of each Vocal Cords's Gastrointestinal Smooth Muscle Cells equals 4. How many Cells does Respiratory Mucosa have?

► **Responses + Labels:**
"Define Vocal Cords's Gastrointestinal Smooth Muscle Cells as W;" [+]
"so W = 4." [+]
"Define Vocal Cords's Arrector Pili Muscle Cells as p;" [−]
"so p = 3 * W = 3 * 4 = 2." [−]
"Define Nasal Cavity's Gastrointestinal Smooth Muscle Cells as g;" [−]
"so g = 3 * p = 3 * 2 = 1." [−]
"Define Nasal Cavity's Arrector Pili Muscle Cells as e;" [−]
"c = g - p = 1 - 2 = 4;" [−]
"so e = 3 + c = 3 + 4 = 2." [−]
"Define Respiratory Mucosa's Pericytes as z;" [−]
"w = g + p = 1 + 2 = 3;" [−]
"so z = 4 * w = 4 * 3 = 2." [−]
"Define Respiratory Mucosa's Arrector Pili Muscle Cells as F;" [−]
"so F = 2." [−]
"Define Respiratory Mucosa's Gastrointestinal Smooth Muscle Cells as P;"" [−]
so P = e = 2." [−]
"Define Respiratory Mucosa's Cells as m;" [−]
"G = P + F = 2 + 2 = 4;" [−]
"so m = G + z = 4 + 2 = 1." [−]
► **Parser:**
existing_but_not_required_params: Vocal Cords's Arrector Pili Muscle Cells

Thus, this allows us to control the amount of IC present and ultimately measure the effects of unnecessary information on LLM's responses. We prepared a dataset of 5000 problems of varying OP values $\in$ [2, 15] as well as an additional 1000 problems of OP=15 so that the model had a sufficient number of high operation training problems. Ultimately, the PRM was trained on each problem and each of its steps and used to facilitate our Tree of Thoughts (ToT) algorithm.

In settings with a PRM, we generated responses in a step by step manner by using ";" and "." as our intermediary stop tokens. Each intermediary step would be scored by the PRM and only the top N/M responses would be selected as candidates in the next step to be explored further. This process was repeated until the LLM generated the <*EOS*> token, signaling that the response was complete. This final response would then be passed into the parser to determine its correctness.

E. Finetuning Details

Model We finetune **LLaMA 3.2-1B Instruct** model released by Meta using both LoRA finetuning and continued pretraining (full finetuning). This model adopts a decoder-only transformer architecture with rotary positional embeddings and gated MLP layers. All experiments are performed on two NVIDIA H100 GPUs.

Finetuning Configuration Due to some computational constraints, our training is conducted on a fixed dataset of 30,000 samples. Each example contains a complete problem-solution pair, and inputs exceeding 2048 tokens are filtered out. We use a context length of 2048, a learning rate of 5e−5, and the AdamW optimizer with cosine learning rate decay. Training proceeds for 50 epochs with a batch size of 8 and gradient accumulation of 8 steps, yielding an effective batch size of 64. We apply mixed-precision training with `bfloat16`, no warmup, and a maximum gradient norm of 1.0. Flash attention is enabled.

Evaluation Protocol We evaluate each model on a fixed test set containing 100 examples per reasoning operation and per level of irrelevant context (IC), including *Light*, *Medium*, and *Hard*. Evaluations are performed separately on in-distribution (ID) and out-of-distribution (OOD) data. This setup enables precise measurement of reasoning robustness under varying levels of distractibility, supporting our core analysis of how irrelevant information affects model behavior.

Architectural Generalization. Recent controlled studies suggest that decoder-only transformer models equipped with full attention and rotary positional embeddings tend to exhibit similar learning dynamics and inductive biases, even when implemented under different architectures. These models—such as GPT-style, LLaMA, Mixtral, and others—differ in details like normalization placement or gated MLPs, but such variations do not appear to fundamentally alter their learnability or reasoning behavior in practice (Allen-Zhu & Li, 2024a;b;c). In our case, although early experiments were conducted using a LLaMA-style architecture, all final results presented in this paper are based on the more recent *LLaMA 3.2–1B Instruct* model. We did not observe substantial performance differences across architectures during preliminary runs. Given resource constraints, we focus on LLaMA 3.2–1B in this version; however, we acknowledge that running a comprehensive comparison across reasoning models (*e.g.* DeepSeek-R1) would strengthen the generality of our findings and plan to pursue this in future work.

Does More In-Distribution Data Help? To identify an effective training budget, we varied the number of in-distribution samples from 1 K to 30 K and observed saturating OOD gains around 30 K (Figure 9). Based on this, we fixed 30 K samples for all subsequent experiments.

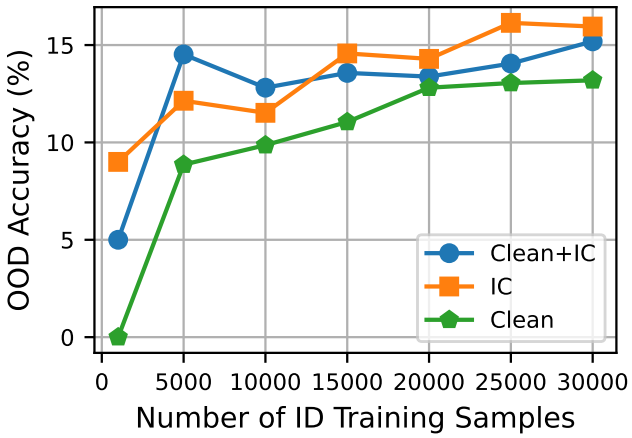

Figure 9: OOD step accuracy as a function of in-distribution training size.