GENERALIZATION OF RLVR USING CAUSAL REASONING AS A TESTBED

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) is increasingly used for post-training large language models (LLMs) to reason, but it remains unclear when RLVR yields reliable generalization. This paper investigates the generalization of RLVR using causal reasoning problems as a testbed, namely probabilistic inference in a causal graphical model. We choose this setting because causality is an important area that LLMs still struggle with, and because this setting provides us with two natural axes of difficulty along which to systematically probe generalization: the level of the probabilistic query—associational, interventional, and counterfactual—and the complexity of the query, as measured by the size of its relevant subgraph. We generate datasets of causal graphs and queries spanning these axes of difficulty and use them to fine-tune Qwen-2.5-Instruct models using RLVR and SFT, varying the query level seen during training and the model scale (3B–32B). Our experiments show that RLVR achieves stronger within- and across-level generalization than SFT, but only on a subset of (model scale, query level) configurations. We trace the source of RLVR's effectiveness (or lack thereof) partly to the reasoning capability of the LLM on a particular level prior to fine-tuning. RLVR then improves the marginalization strategy and reduces probability derivation errors in the reasoning steps, significantly boosting accuracy overall and especially on more complex queries. Overall, we found RLVR significantly improves generalization on casual reasoning queries at the association and intervention level, but counterfactual level queries remain challenging for all models investigated in our experiments.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) (Lambert et al., 2025; DeepSeek-AI et al., 2025) is increasingly being used in post-training large language models (LLMs) to reason, leveraging supervision from domains where answer correctness can be automatically checked with verifiers, such as math problem-solving (Shao et al., 2024; Lambert et al., 2025; DeepSeek-AI et al., 2025), theorem proving (Xin et al., 2024; Ren et al., 2025; Wang et al., 2025), and code generation (Le et al., 2022; Liu & Zhang, 2025). However, it remains unclear when LLMs fine-tuned with RLVR acquire reliable generalizations beyond their training data.

A recent line of work starts to explore this question by comparing the generalization of reinforcement learning fine-tuning (RL) with supervised fine-tuning (SFT) or their combination (Chu et al., 2025; Chen et al., 2025; Swamy et al., 2025; Qiu et al., 2025). Particularly relevant to our work is Chu et al. (2025), which studies the generalization of RLVR/SFT to novel variants of text and visual reasoning tasks. This paper also investigates the generalization of RLVR, and compares with a SFT baseline. However, we focus on identifying situations in which RLVR itself generalizes effectively (versus not), and we focus on the causal reasoning domain.

Causal reasoning is useful for systematically probing the generalization of RLVR—it distinguishes three distinct levels of reasoning: association, intervention, and counterfactual (a.k.a the ladder of causation Bareinboim et al., 2022; Pearl & Mackenzie, 2018), with which we can probe generalization both within- and across-level. Causal reasoning with LLMs is also itself a topic with growing interest, with studies evaluating the causal knowledge and reasoning capabilities of existing LLMs

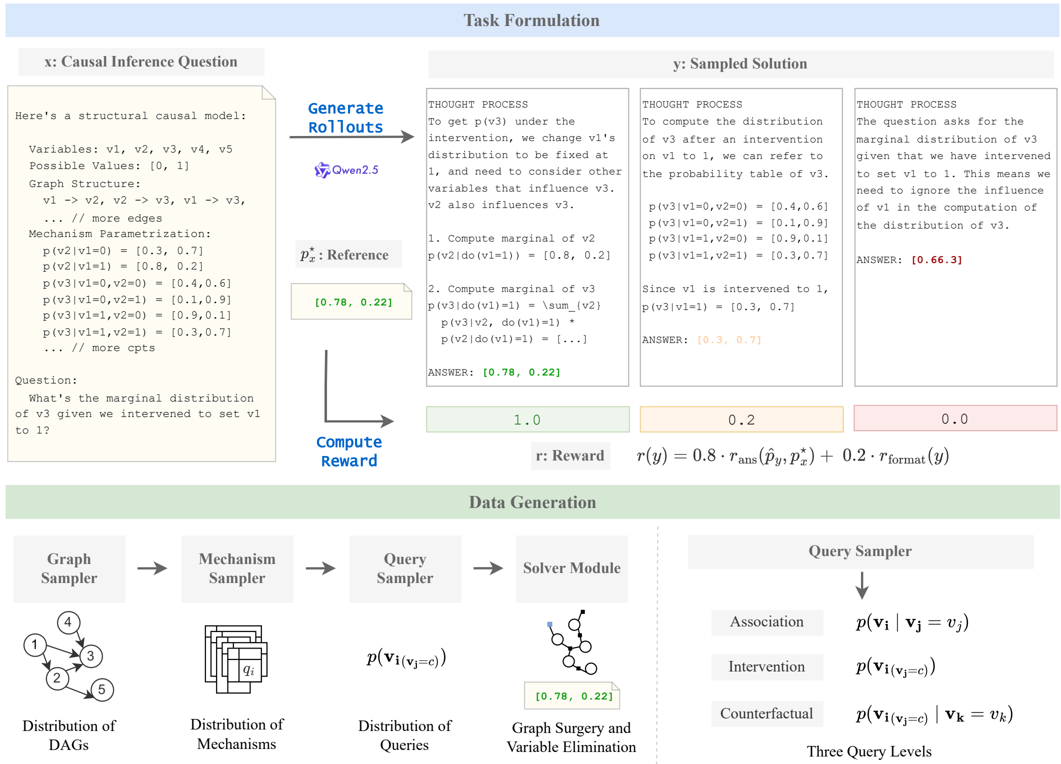

Figure 1: Top: Our causal inference task for investigating generalization of RLVR (see section 2), system prompt (fig. 8) omitted for space. Bottom Left: Generative process for sampling task instances, and solver for computing the reference (see section 3). Bottom Right: We generate association, intervention, and counterfactual queries to study RLVR's within-/across-level generalization.[2]

in both commonsense settings (Zečević et al., 2023; Kıcıman et al., 2024; Chi et al., 2024; Yu et al., 2025a) and formal settings (Jin et al., 2023; 2024; Yu et al., 2025a).

In this paper, we focus on the formal task of probabilistic inference in a causal graphical model (fig. 1 top). Concretely, our task input x contains a description of a causal graph and a query. For RLVR, the LLM outputs intermediate reasoning followed by a probability distribution $\hat{p}$ as the final answer.[1] For SFT, the LLM directly outputs the distribution $\hat{p}$. We define processes for sampling instances of our task, and compute for each instance a reference answer $p^\star$ via variable elimination.

We then run RLVR and SFT fine-tuning experiments starting from a representative LLM family, Qwen2.5-Instruct (Qwen et al., 2025), varying *model size* and *the level of queries* seen during training. Our RLVR training uses variants of GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025b), and our SFT baseline is trained to maximizes the probability of $p^\star$ conditioned on task input x.

We present the following findings from our experiments and analysis:

1. **Within- and Across-level Generalization** When trained and evaluated on the same query level, RLVR achieves stronger generalization than SFT on association and intervention queries for models $\geq$ 7B, but under-performs on 3B (for all levels) and counterfactual level (for all sizes); When measuring generalization to a different query level from training,

[1]The reader would be correct to point out that the causal ladder, contrasts the different *knowledge* required to answer each level of questions (Bareinboim et al., 2022), but our setting with full SCM parametrization as input, eliminates this difference. However, queries from each of levels still need different modes of reasoning—abduction for association, deduction for intervention, and abduction followed by deduction for counterfactual. We discuss in section 3 how our setting affects the difficulty ordering of the three levels.

[2]We write intervention queries as $p(\mathrm{v}_{i\,(\mathrm{v}_j=c)})$ for consistency in notation with the counterfactual queries. It is equivalent to $p(\mathrm{v}_i \mid do(\mathrm{v}_j = c))$ for readers more familiar with the *do* notation (Pearl, 2009).

RLVR outperforms SFT on sizes $\geq$ 7B (figs. 3 and 4). RLVR is often more precise than SFT and better on more complex queries (fig. 6).

2. **Scaling and LLM's Reasoning Prior** We trace the effectiveness of RLVR partly back to the strength of the LLM reasoning prior to fine-tuning. Scaling up the size of the LLM improves reasoning significantly: 3B models fail to reason before and after RLVR, while 32B models with *zero-shot* reasoning beats its counterpart *after* SFT (fig. 4 bottom).
3. **What did RLVR learn?** Overall, RLVR builds on top of the reasoning prior by fixing probability derivation errors (e.g. dropping dependencies) and by learning a incremental marginalization strategy that systematically sums out relevant variables (fig. 5).

Overall, our findings contribute to the understanding of RLVR's generalization behavior as well as its effectiveness on enhancing LLMs' reasoning capabilities on formal causal reasoning tasks.

2 METHOD

We are interested in studying the limits of generalization of RLVR using the task of probabilistic inference in causal graphical models. In this section, we discuss the task definition (section 2.1), training objectives (section 2.2), and the main factors we will vary in our experiments (section 2.3).

2.1 TASK DEFINITION

Please refer to fig. 1 (top) for an illustration of the task input and output.

Input We use Qwen2.5-Instruct (Qwen et al., 2025) as our base model. The input x for our task consists of a system message x_{sys} containing short instructions for task and format (fig. 8, appendix) and a user message x_{user} describing a concrete task instance (fig. 11, appendix), including a description of the causal graphical model and a query. The description of the causal graph includes variable definitions, the graph structure, and mechanism parametrizations.

Output For RLVR, the output y is a reasoning chain followed by a probability distribution. For SFT, the output y is directly a probability distribution. See fig. 1 (top) for an illustration of the task for RLVR. We perform extraction of the answer $\hat{p}_y$ from y using regular expression. Each training instance x comes with a reference answer $p^\star_x$.

2.2 TRAINING OBJECTIVES

Reinforcement Learning with Verifiable Rewards We optimize the RL objective $\mathbb{E}_{x\sim T}\mathbb{E}_{y\sim p_\theta(x)}\left[r(y)\right]$, where T is the distribution over training instances. Following typical RLVR setups (DeepSeek-AI et al., 2025), we use a combination of format and accuracy reward, specifically $r(y) = 0.8 \cdot r_{\text{ans}}(\hat{p}_y, p^\star_x) + 0.2 \cdot r_{\text{format}}(y)$, where

$$r_{\text{ans}}(p, q) = \mathbf{1}[D(p, q) < 0.01] \quad r_{\text{format}}(y) = \mathbf{1}[\hat{p}_y \text{ can be extracted}] \tag{1}$$

and $D(p, q) := \frac{1}{2}\int_x p(x) - q(x)dx$ is the total variation distance. We round $\hat{p}_y$ and $p^\star_x$ to the nearest two decimal points and use $t = 0.01$, where r_{ans} effectively measures exact match accuracy.

Supervised Fine-tuning For our supervised fine-tuning baseline, we directly maximize the conditional likelihood of the reference answer $y^\star_x$ for input x, $\mathbb{E}_{x\sim D} \log p_\theta(y^\star_x \mid x)$.

2.3 STUDY DESIGN

Data Setup We stratify the task instances across two axes of difficulty: first, by the query's level (association, intervention, or counterfactual), and second, by the complexity of the query as measured by its relevant subgraph.[3] We vary the source of the training data between individual levels to measure across-level generalization, and breakdown within-level generalization by complexity.

Fine-tuning Setup We vary the model size between 3B, 7B, and 32B within the Qwen2.5-Instruct family, using variants of GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025b) for RL, and maximum likelihood of $p^\star$ conditioned on x for SFT.

[3]Details on this relevant subgraph metric are in section 3.

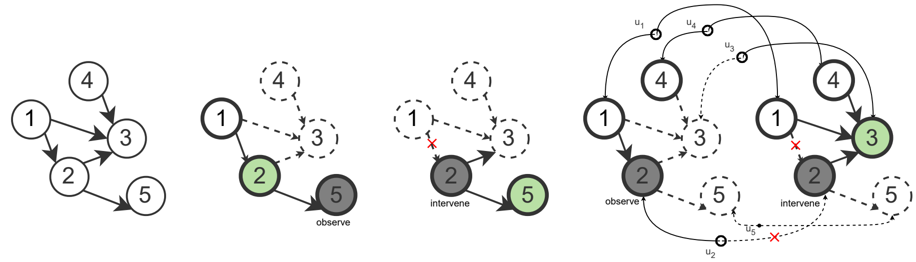

Figure 2: Illustration of graph modifications corresponding to each query level and its relevant (solid) and irrelevant subgraph (dashed). × denotes dependencies removed due to an intervention. Relevant nodes are defined as ancestors of either the observation or the query variable, after graph modifications are performed to account for any interventions. Left: original graph. Mid-left: 3 relevant nodes for association query $p(\mathrm{v}_2 \mid \mathrm{v}_5 = v_5)$. Mid-right: 2 relevant nodes for intervention query $p(\mathrm{v}_{5\,(\mathrm{v}_2=c)})$. Right: 11 relevant nodes for counterfactual query $p(\mathrm{v}_{3\,(\mathrm{v}_2=c)} \mid \mathrm{v}_2 = v_2)$.

3 Data Generation

In fig. 1 (bottom left) we provide a diagram for the generative process of synthesizing data for our task. We first describe the objects that we sample, then the step by step process for sampling them.

Structural Causal Models We use structural causal models (SCMs) (Pearl et al., 2016) with binary variables,[4] no cycles, and independent noise variables as our causal graphical model family. Such a structural causal model $M = (G, \mathbb{F}, \mathbb{Q})$ is defined by a DAG $G = (V, E)$, where each node $i \in V$ is associated with a variable v_i and a *deterministic* function $f_i \in \mathbb{F}$ that defines its relationship with its parents $\mathrm{pa}(\mathrm{v}_i)$ and a noise variable u_i following distribution $q_i \in \mathbb{Q}$. This gives

$$v_i := f_i(\mathrm{pa}(\mathrm{v}_i), u_i), u_i \sim q_i \tag{2}$$

which induces conditional distributions $p(\mathrm{v}_i \mid \mathrm{pa}(\mathrm{v}_i))$ for all variables v_i.

Queries fig. 2 illustrates queries of each level and their graph modifications. Following notation in Pearl et al. (2016), an *association level* query $p(\mathrm{v}_i \mid \mathrm{v}_j = v_j)$ concerns statistical dependence, asking about the distribution of v_i given an observed value $\mathrm{v}_j = v_j$. An *intervention level* query $p(\mathrm{v}_{i\,(\mathrm{v}_j=c)})$ concerns causal effects, asking about the distribution of v_i under an external intervention that sets v_j to c. A *counterfactual level* query $p(\mathrm{v}_{i\,(\mathrm{v}_j=c)} \mid \mathrm{v}_k = v_k)$ concerns hypothetical alternatives, asking about the distribution of v_i had v_j been set to c, in a world where we in fact observed $\mathrm{v}_k = v_k$.[5]

D1: Graph Sampler The first step of sampling a SCM is sampling a DAG g. Given the desired size N, we adopt Lampinen et al. (2023)'s workflow to generate a graph, which first samples a random number of independent nodes between 1 and N. Additional nodes are then introduced iteratively until we reach N total nodes. Each added node has either one or two parents chosen uniformly from the existing nodes. Finally, the nodes are renamed with a random permutation.

D2: Mechanism Sampler For each v_i and for each joint assignment $\mathbf{v}$ to its parents $\mathrm{pa}(\mathrm{v}_i)$, we sample a binary distribution $q_{\mathbf{v}}$ uniformly from the simplex. We then define one noise variable $\mathrm{u}_i^{\mathbf{v}} \sim q_{\mathbf{v}}$ per each $\mathbf{v}$, and define the mechanism f_i to simply select one particular noise variable's value to take on based on $\mathbf{v}$, namely $v_i = f_i(\mathbf{v}, \boldsymbol{u}_i) = u_i^{\mathbf{v}}$. This simple mechanism directly maps the noise distributions $q_{\mathbf{v}}$ onto rows of the conditional probability table $p(\mathrm{v}_i \mid \mathrm{pa}(\mathrm{v}_i))$.

D3: Query Sampler Given a SCM, we then sample queries for a chosen level. Association level queries contain one observation, intervention level one intervention, and counterfactual level one of each (fig. 2). We sample the target variable v_i uniformly. For association and intervention level queries, we also uniformly sample a variable v_j to condition on or intervene on, respectively. For the counterfactual query, we sample the intervention variable v_j first, and then sample the observation v_k from its descendants. Observations are drawn from the SCM, interventions are uniform $\{0, 1\}$.

[4] We choose binary variables for speed of computing the ground-truth solution—exact inference in graphical models is NP-hard in general, and slows down significantly in practice with cardinality and graph size.

[5] The observations/interventions in our data are always on a single variable for simplicity, and LLMs already struggle in this simple setting. Future work could explore vector-valued observations and interventions.

D4: Solver Given the full specification of a SCM $M = (G, \mathbb{F}, \mathbb{Q})$ and a query q of association, intervention or counterfactual level, we reduce it to exact inference of some q' in a *possibly modified* SCM $M' = (G', \mathbb{F}', \mathbb{Q})$. We then use variable elimination (Zhang & Poole, 1994) to compute the answer. The modified graphs are illustrated in fig. 2, with additional details in appendix A.1.

Difficulty Metric We first stratify queries by their *level*, as they represent different modes of reasoning. In our setting where we provide the fully parametrized SCM as input, association queries of the form $p(\mathrm{v}_i \mid \mathrm{v}_j = v_j)$ requires abduction (summing out ancestors in the posterior), intervention queries of the form $p(\mathrm{v}_{i(\mathrm{v}_j=c)})$ requires deduction (sum out ancestors after fixing v_j), and counterfactual requires abduction followed by deduction (infer posterior of noise variables, then sum out noise variables and ancestors in an alternative world after fixing v_j). This changes the difficulty ordering from the usual association $<$ intervention in the causal ladder to association $>$ intervention in our setting, since computing posterior given v_j usually requires more work than fixing v_j at c.[6]

Within each level, we also measure difficulty by $|V_{\text{rel}}|$, the complexity of the query, defined as the size (number of nodes) of the subgraph relevant to the query. Relevant nodes are ancestors of either the observed variable, or the query variable, in the modified graph G'. Irrelevant nodes in $V' \setminus V_{\text{rel}}$ are descendants of V_{rel}, and their factors sums out to 1 during variable elimination and can be ignored. See fig. 2 for example relevant subgraphs for each level.

4 Experiments

4.1 Experiment Setup

Dataset Construction For each level in $\{\text{association}, \text{intervention}, \text{counterfactual}\}$, we generate a training, development, and test set, consisting of 8000, 2000, and 8000 examples respectively. Each example is a query over a parametrized causal graph over 10 binary variables. We ensure that the SCMs in training, development and test sets are disjoint. Refer to appendix A.2 for more details.

$|V_{\text{rel}}|$ Distribution See fig. 7 (appendix) for histograms of query complexity metric $|V_{\text{rel}}|$. When presenting results, we group examples by ranges of $|V_{\text{rel}}|$ with the following cutoffs: 1-3 (small), 4-6 (medium), 7-10 (large) for association, 1-2 (small), 3-4 (medium), 5-10 (large) for intervention, and 1-7 (small), 8-15 (medium), 16-30 (large) for counterfactual.[7] The complexities are meant to be used within-level and is generally *not* comparable across levels.

Metrics Given a input x for a language model, its reference solution $p^\star_x$, and a LLM output y, we measure its correctness by the following metric based on total variation distance and a format requirement. Let $\hat{p}_y$ be a solution extracted from y,

$$\textsc{Correct}_t(x, y) := \begin{cases} 0 & \text{if format error, failed to extract } \hat{p}_y \\ 1 & \text{if } D(\hat{p}_y, p^\star_y) < t \end{cases} \tag{3}$$

where $D(p, q) := \frac{1}{2}\int_x p(x) - q(x)dx$ is the total variation distance. We round $\hat{p}_y$ and $p^\star_x$ to the nearest two digits and use $t = 0.01$, which effectively measures exact match accuracy.

Filtering Due to rounding, on many instances the intervention and observation may not result in a measurable change in the marginal distribution of the query variable, which can introduce false positives for evaluation. Therefore, we filter out such examples in our main analysis, but also include the unfiltered results in appendix D.

Fine-tuning And Inference Setup **For RLVR**, We use GRPO with the token-level normalization, and DAPO without the overlong buffer for simplicity. We use a batch size of 8, with 32 roll-outs per example, a learning rate of 10^{-6}, and other hyperparameters default from DAPO implemented in the VERL library (Sheng et al., 2024). We train for 7.5k steps for 3B and 7B models, and 2.5k steps for 32B models, as they are roughly a third slower to train compared to 7B. **For SFT**, We train using maximum likelihood on $p^\star$ for 5k steps, with learning rate 10^{-6} and pick best checkpoint (saved every 200 step) by picking best loss on development set. **For inference**, we decode at temperature 0. Additional details on hyper-parameters are included in appendix B.1.

[6] If we also included intervention queries with *conditions*, then it would require graph modification followed by abduction, and the ordering would be reverted back to the usual association $<$ intervention.

[7] Twin network graphs (Shpitser & Pearl, 2012) increases the number of explicitly represented nodes in G.

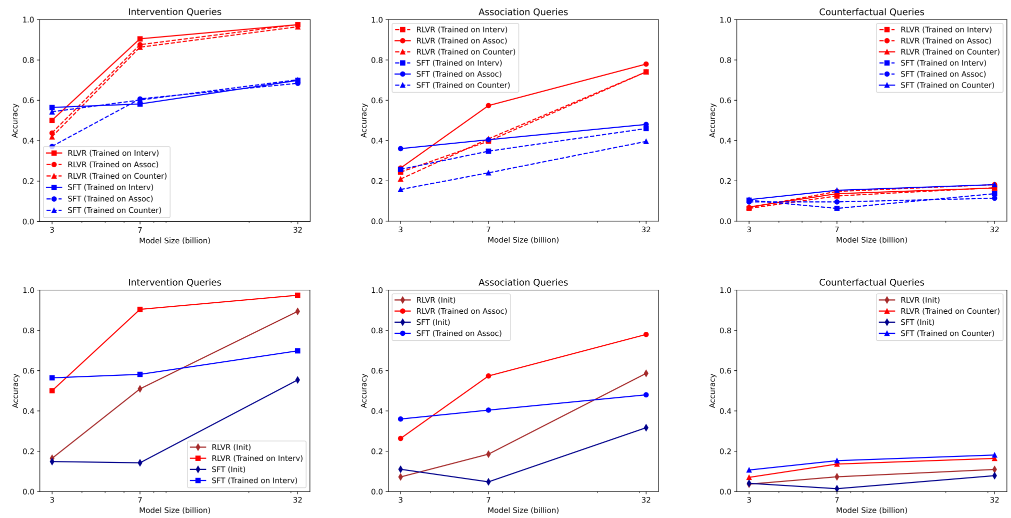

Figure 4: Top: Accuracy (y-axis) vs. LLM size (x-axis) when evaluated on intervention (left), association (middle), and counterfactual (right) queries. Red curves correspond to RLVR, blue curves correspond to SFT. Solid (−) curves are LLMs fine-tuned on the *same* level as evaluation, dashed (−−) curves are trained on a *different* level from evaluation. Bottom: Reasoning (RLVR) vs non-reasoning (SFT) strategies, before and after fine-tuning. As scale increases, both reasoning and non-reasoning prior improve, though the reasoning prior benefits more from scaling.

4.2 Main Results

We focus our discussion and analysis on GRPO as the representative RLVR algorithm in the main text, as it is simple, widely used, and its results are not significantly different from DAPO in our experiments. DAPO results are included in appendix D.

In fig. 4, we compare the accuracy of LLMs fine-tuned via RLVR and SFT on the three query types—intervention, association, and counterfactual. We vary the model size between 3B and 32B, and we vary the query type that the model was fine-tuned on.

Within-level Generalization: RLVR outperforms SFT on only a subset of (model size, query type) configurations. In fig. 3 left, we show the size and query type configurations for which RLVR outperforms SFT, when trained and evaluated on the same query type —RLVR significantly outperforms SFT on intervention and association queries, for sizes $\geq$ 7B.

Within-level

	Interv	Assoc	Counter
3B	**SFT**	**SFT**	**SFT**
7B	**RLVR**	**RLVR**	SFT
32B	**RLVR**	**RLVR**	SFT

Across-level

	Interv	Assoc	Counter
3B	**SFT**	**SFT**	**SFT**
7B	**RLVR**	**RLVR**	**RLVR**
32B	**RLVR**	**RLVR**	**RLVR**

Figure 3: The algorithm with higher within-/across-level accuracy for different sizes and query types. Significant cells (paired-perm test at $p < 0.05$) bolded and colored.

Across-level Generalization: RLVR outperforms SFT beyond 3B models. Larger LLMs perform better across levels for both RLVR and SFT. In fig. 3 right, we see when evaluating on different level from training, RLVR outperforms SFT on models $\geq$ 7B. In fig. 4 we find that for both SFT and RLVR, the performance gap between LLMs fine-tuned on in- and out-of-level queries generally decreases as model scale increases, suggesting better cross-level generalization with scaling.

4.3 Analysis

Analysis-I: RLVR is ineffective when the reasoning capability of the base model is too poor prior to fine-tuning. Why is RLVR ineffective on 3B and counterfactual queries, as seen in fig. 3?

3B models attempt explicit marginalization before fine-tuning, but succeed rarely; After fine-tuning it outputs answer directly without explicit marginalization. We reviewed a subset of reasoning traces across levels and sizes, before and after RLVR fine-tuning. All models attempt explicit marginal-

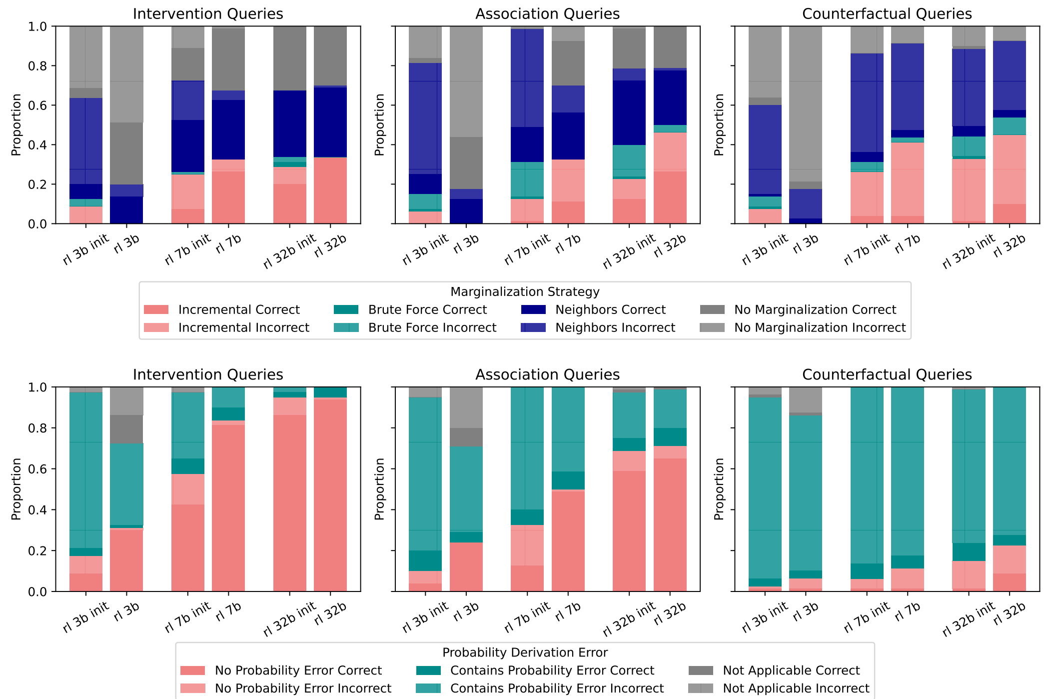

Figure 5: LLM judge (o4-mini) analysis of reasoning strategy (top) and existence of derivation errors (bottom) before and after RLVR. Marginalization strategies are annotated on 80 samples per level. Derivation errors are annotated on the same samples. Judge prompts (including category definitions) are included in fig. 10. Example traces of each strategy are included in figs. 12 to 15.

ization before fine-tuning, but only models $\geq$ 7B continue attempting explicit marginalization after fine-tuning. Our analysis of reasoning traces in fig. 5 show that at 3B, traces that attempt to marginalize step by step (incremental, brute force, and neighbors) are rarely correct—a possible explanation for their regression to directly predicting the answer after fine-tuning.

On counterfactual level queries, models never attempted to build twin-networks or perform inference over exogenous variables, both before and after RLVR. We reviewed a subset reasoning traces from models of different sizes, but did not observe any attempts to create a twin-network-graph. We conducted an oracle experiment where additional hints on solving the counterfactual query via twin network graph is provided in the system prompt (fig. 9, appendix). However, its accuracy is not very different from the original prompt without hints.[8] This result is in contrast to more positive findings in previous evaluations of LLMs' counterfactual reasoning in the commonsense settings (Kıcıman et al., 2024) or formal settings with continuous mechanisms(Tu et al., 2024). This contrast may be partly due to our strict exact-match metric and more numerically challenging task.

Overall, these findings suggest that RLVR is sensitive to the LLM's reasoning success rate prior to fine-tuning, echoing the cold start problem (DeepSeek-AI et al., 2025) in RLVR post-training. Since we start from Qwen2.5-Instruct (Qwen et al., 2025), which is already instruction-tuned and has some reasoning-tuning too, the limitations seen here is likely more specific to the causal domain.

Analysis-II: LLM's success rate prior to fine-tuning improves with scale and especially with reasoning. The prior is a major source of RLVR's effectiveness. In fig. 4 (bottom), we show the initial accuracy of LLMs prompted to reason (used in RLVR) versus LLMs prompted to directly predict the answer (used in SFT). Across all levels, both the reasoning prompt and the direct prediction prompt's accuracy prior to fine-tuning increases substantially with scale, but reasoning benefits more from scaling. Furthermore, the reasoning prompt prior to fine-tuning achieves a higher accuracy than non-reasoning prompt across the board. Notably, on intervention and association queries,

[8] See fig. 16 in the appendix for results on the oracle experiment and fig. 9 for system prompt with hint.

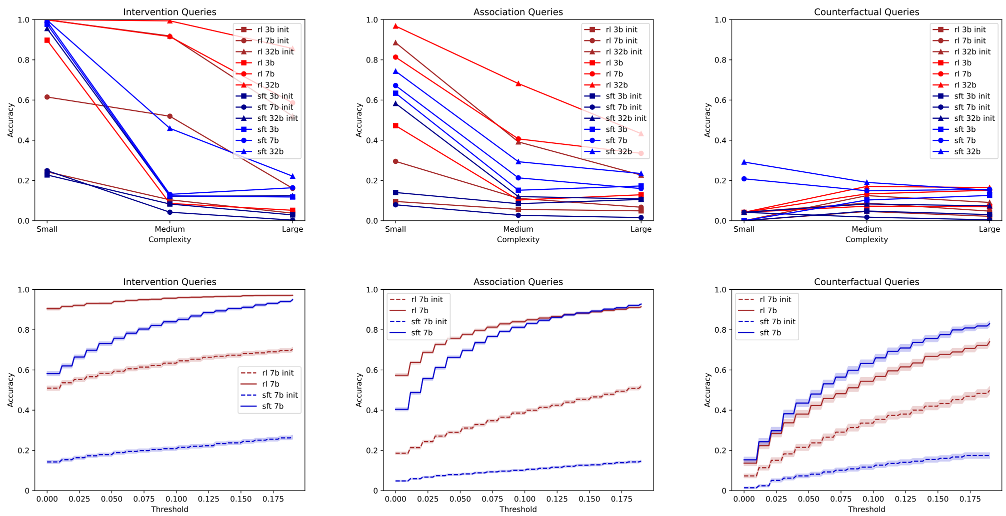

Figure 6: Top: Within each query level, accuracy vs. query complexity V_{rel} increases. Note that complexities are not comparable across levels. Bottom: Accuracy as the threshold for correctness $t \in (0.01, 0.2]$ is relaxed for 7B models. x-axis plots threshold for accuracy t (the lower the stricter), and y-axis plots accuracy at t. Across all levels (left to right), we see that RL models are often more precise than SFT models. See figs. 19 to 21 for the same plot but for all query levels and model sizes. The observable staircase pattern is due to rounding to two digits.

32B models fine-tuned with SFT still under-performs zero-shot reasoning (fig. 4 bottom). This demonstrates the substantial benefit of the reasoning prior, which we next show RLVR improves on.

Analysis-III: What did LLM learn through RLVR? We manually examined a subset of reasoning traces across all levels before and after RLVR. Qualitatively, models $\geq$ 7B often take a incremental approach to marginalization, summing out variables locally following the graph structure (e.g. going from leaves to root), while taking observations and interventions into account. On the other hand, 3B models learn to do some thinking then directly predict the answer without calculations. In terms of error sources, reasoning traces often contain numeric errors, copy paste errors, but most destructively, probability derivation errors such as missing dependencies, misinterpreting question, or forgetting to sum over certain variables.

To understand in more detail the distribution of marginalization strategies and reasoning errors before and after RLVR fine-tuning, we prompted o4-mini to annotate the marginalization strategies (fig. 5 top), and the existence of derivation errors (fig. 5 bottom) in reasoning traces. The prompts are included in fig. 10 (appendix). The first author validated the prompt with 10 samples per category for the strategy (37/40 agreement) and error annotations (25/30 agreement). Please refer to appendix for sample traces and tabular results behind fig. 5.

Overall, we see that RLVR fine-tuning significantly reduces probability derivation errors and such errors correlate highly with incorrectness (fig. 5 bottom). RLVR also shifts the distribution over marginalization strategies towards an incremental marginalization strategy, and especially so on the more complex queries (see fig. 18 top in appendix). Incremental marginalization solutions systematically follow the graph structure to identify variables that needs to be summed over, and then writes formulas to marginalize them out one after another. Given these improvements, we expect to see generalization to improve on larger queries, which we observe in fig. 6 (top). These improvements also likely contribute to the improvement in overall performance both within-level and across-level fig. 4, as these skills (incremental marginalization, and correct probability derivations) would be generally useful across levels.

Analysis-IV: How does RLVR models behave differently from SFT models? In fig. 6 (top), we show that within each query type, on 7B and 32B LLMs, SFT tend to perform better on the less complex queries of that type, while RLVR performs better on more complex queries of that type.

In fig. 6 (bottom), we plot the proportion of correct answers as we relax the criterion of correctness from exact match to within 0.2 in total variation distance. We show that LLMs fine-tuned with RLVR are more precise, while SFT models often is only able to get the solution approximately correct. Both RLVR and SFT significantly improve the precision of the LLMs relative to base.

5 Related Work

RLVR for LLM Post-Training Reinforcement learning is becoming a common step of LLM post-training. Early works use RL to align LLMs using human preference data (Ouyang et al., 2022). Reinforcement learning with verifiable rewards (RLVR) (Lambert et al., 2025; DeepSeek-AI et al., 2025) teach reasoning on domains where reward (often a combination of correctness and format) can be verified automatically. This led to significant progress in domains such as math problem solving theorem proving, and code generation (DeepSeek-AI et al., 2025; Ren et al., 2025; Liu & Zhang, 2025). In this paper, we attempt to understand the successes and limitations of RLVR's generalization, using a controlled family of causal reasoning tasks.

Understanding RLVR's Generalization Several recent studies explored the respective contribution of SFT and RL to the *generalization* of the resulting LLM (Chu et al., 2025; Chen et al., 2025; Swamy et al., 2025), with evidence that SFT is useful for warm-starting, while RL can significantly improve generalization. Another line of work investigates how SFT and RL should be ordered or combined to achieve the best generalization result (Liu et al., 2025; Qiu et al., 2025). Yet another line of work studies the sources of RLVR's effectiveness and where it improves models (Swamy et al., 2025; Qin et al., 2025). Similar to the earlier works, we investigate generalization of RLVR and SFT, and similar to the latter works, we focus on understanding the limits and the sources of RL's effectiveness itself. Different from these works, we chose the causal reasoning domain, a less explored but important area that we show remains challenging.

LLM for Causal Reasoning LLMs' understanding of causality has been receiving increasing interest (Yu et al., 2025a), as LLMs make their way into domains like medicine and law where causal reasoning is important. Prior work investigates the commonsense causal knowledge and reasoning of pretrained LLMs (Kıcıman et al., 2024; Zečević et al., 2023; Chi et al., 2024), or their performance under more formal settings (Jin et al., 2023; 2024; Tu et al., 2024). Our work focuses on the more formal cases, namely a challenging task of solving numeric probabilistic queries in causal graphical models, systematically investigating the generalization of different fine-tuning methods.

6 Conclusion and Future Directions

In this paper, we investigated the generalization behavior of RLVR and SFT using probabilistic inference in causal graphical models as a testbed. Our main findings are that RLVR is less effective when the task is too challenging for the model prior to fine-tuning, but when the LLM has a basic level of successful reasoning rate on a problem level, RLVR significantly improves generalization, by fixing domain-specific reasoning errors and boosting systematic marginalization strategies, outperforming SFT, and especially on more complex queries. We also find that the LLM's prior (both reasoning and direct prediction) scales with model size, and scaling gains are stronger with reasoning.

Our findings add to an emerging line of investigation into RLVR's generalization (Chu et al., 2025; Chen et al., 2025; Swamy et al., 2025). A future direction is to explore more deeply the roles of execution quality and strategy quality in reasoning and RLVR (e.g. see recent works Qin et al., 2025; Sinha et al., 2025). This may provide us more insights into the mechanism of RLVR's generalization.

Our findings also add to an increasing body of work studying LLMs for causal reasoning (Kıcıman et al., 2024; Jin et al., 2023; 2024). Their effectiveness on some commonsense counterfactual causal reasoning settings (Kıcıman et al., 2024) contrasts with our results on more challenging formal counterfactual reasoning problems. Future work could possibly bridge this gap with RLVR, where the formal setting could benefit from heuristics acquired from training on informal causal reasoning examples, and the commonsense setting could benefit from learning more structured reasoning patterns from synthetic formal data.

REFERENCES

Elias Bareinboim, Juan D Correa, Duligur Ibeling, and Thomas F. Icard. On pearl's hierarchy and the foundations of causal inference. *Probabilistic and Causal Inference*, 2022. URL https://api.semanticscholar.org/CorpusID:232379651.

Hardy Chen, Haoqin Tu, Fali Wang, Hui Liu, Xianfeng Tang, Xinya Du, Yuyin Zhou, and Cihang Xie. Sft or rl? an early investigation into training r1-like reasoning large vision-language models, 2025. URL https://arxiv.org/abs/2504.11468.

Haoang Chi, He Li, Wenjing Yang, Feng Liu, Long Lan, Xiaoguang Ren, Tongliang Liu, and Bo Han. Unveiling causal reasoning in large language models: Reality or mirage? In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 96640–96670. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/af2bb2b2280d36f8842e440b4e275152-Paper-Conference.pdf.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=dYur3yabMj.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Zhijing Jin, Yuen Chen, Felix Leeb, Luigi Gresele, Ojasv Kamal, Zhiheng LYU, Kevin Blin, Fernando Gonzalez Adauto, Max Kleiman-Weiner, Mrinmaya Sachan, and Bernhard Schölkopf. CLadder: A benchmark to assess causal reasoning capabilities of language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=e2wtjx0Yqu.

Zhijing Jin, Jiarui Liu, Zhiheng Lyu, Spencer Poff, Mrinmaya Sachan, Rada Mihalcea, Mona Diab, and Bernhard Schölkopf. Can large language models infer causation from correlation?, 2024. URL https://arxiv.org/abs/2306.05836.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality, 2024. URL https://arxiv.org/abs/2305.00050.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

Andrew Lampinen, Stephanie Chan, Ishita Dasgupta, Andrew Nam, and Jane Wang. Passive learning of active causal strategies in agents and language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 1283–1297. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/045c87def0c02e3ad0d3d849766d7f1e-Paper-Conference.pdf.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 21314–21328. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8636419dea1aa9fbd25fc4248e702da4-Paper-Conference.pdf.

Jiawei Liu and Lingming Zhang. Code-r1: Reproducing r1 for code with reliable rewards. 2025.

Mingyang Liu, Gabriele Farina, and Asuman Ozdaglar. Uft: Unifying supervised and reinforcement fine-tuning, 2025. URL https://arxiv.org/abs/2505.16984.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009. ISBN 052189560X.

Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, Inc., USA, 1st edition, 2018. ISBN 046509760X.

Judea Pearl, Madelyn Glymour, and Nicholas P. Jewell. *Causal Inference in Statistics: A Primer*. John Wiley & Sons, 2016.

Tian Qin, Core Francisco Park, Mujin Kwun, Aaron Walsman, Eran Malach, Nikhil Anand, Hidenori Tanaka, and David Alvarez-Melis. Decomposing elements of problem solving: What ”math” does rl teach?, 2025. URL https://arxiv.org/abs/2505.22756.

Haibo Qiu, Xiaohan Lan, Fanfan Liu, Xiaohu Sun, Delian Ruan, Peng Shi, and Lin Ma. Metis-rise: Rl incentivizes and sft enhances multimodal reasoning model learning, 2025. URL https://arxiv.org/abs/2506.13056.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition, 2025. URL https://arxiv.org/abs/2504.21801.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Ilya Shpitser and Judea Pearl. What counterfactuals can be tested, 2012. URL https://arxiv.org/abs/1206.5294.

Akshit Sinha, Arvindh Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. The illusion of diminishing returns: Measuring long horizon execution in llms, 2025. URL https://arxiv.org/abs/2509.09677.

Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J. Andrew Bagnell. All roads lead to likelihood: The value of reinforcement learning in fine-tuning, 2025. URL https://arxiv.org/abs/2503.01067.

Ruibo Tu, Hedvig Kjellström, Gustav Eje Henter, and Cheng Zhang. Carl-gt: Evaluating causal reasoning capabilities of large language models, 2024. URL https://arxiv.org/abs/2412.17970.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, Jianqiao Lu, Hugues de Saxcé, Bolton Bailey, Chendong Song, Chenjun Xiao, Dehao Zhang, Ebony Zhang, Frederick Pu, Han Zhu, Jiawei Liu, Jonas Bayer, Julien Michel, Longhui Yu, Léo Dreyfus-Schmidt, Lewis Tunstall, Luigi Pagani, Moreira Machado, Pauline Bourigault, Ran Wang, Stanislas Polu, Thibaut Barroyer, Wen-Ding Li, Yazhe Niu, Yann Fleureau, Yangyang Hu, Zhouliang Yu, Zihan Wang, Zhilin Yang, Zhengying Liu, and Jia Li. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning, 2025. URL https://arxiv.org/abs/2504.11354.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data, 2024. URL https://arxiv.org/abs/2405.14333.

Longxuan Yu, Delin Chen, Siheng Xiong, Qingyang Wu, Dawei Li, Zhikai Chen, Xiaoze Liu, and Liangming Pan. CausalEval: Towards better causal reasoning in language models. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 12512–12540, Albuquerque, New Mexico, April 2025a. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.622. URL https://aclanthology.org/2025.naacl-long.622/.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025b. URL https://arxiv.org/abs/2503.14476.

Matej Zečević, Moritz Willig, Devendra Singh Dhami, and Kristian Kersting. Causal parrots: Large language models may talk causality but are not causal, 2023. URL https://arxiv.org/abs/2308.13067.

Nevin Lianwen Zhang and David L. Poole. A simple approach to bayesian network computations. 1994. URL https://api.semanticscholar.org/CorpusID:2978086.

A Additional Details on Data

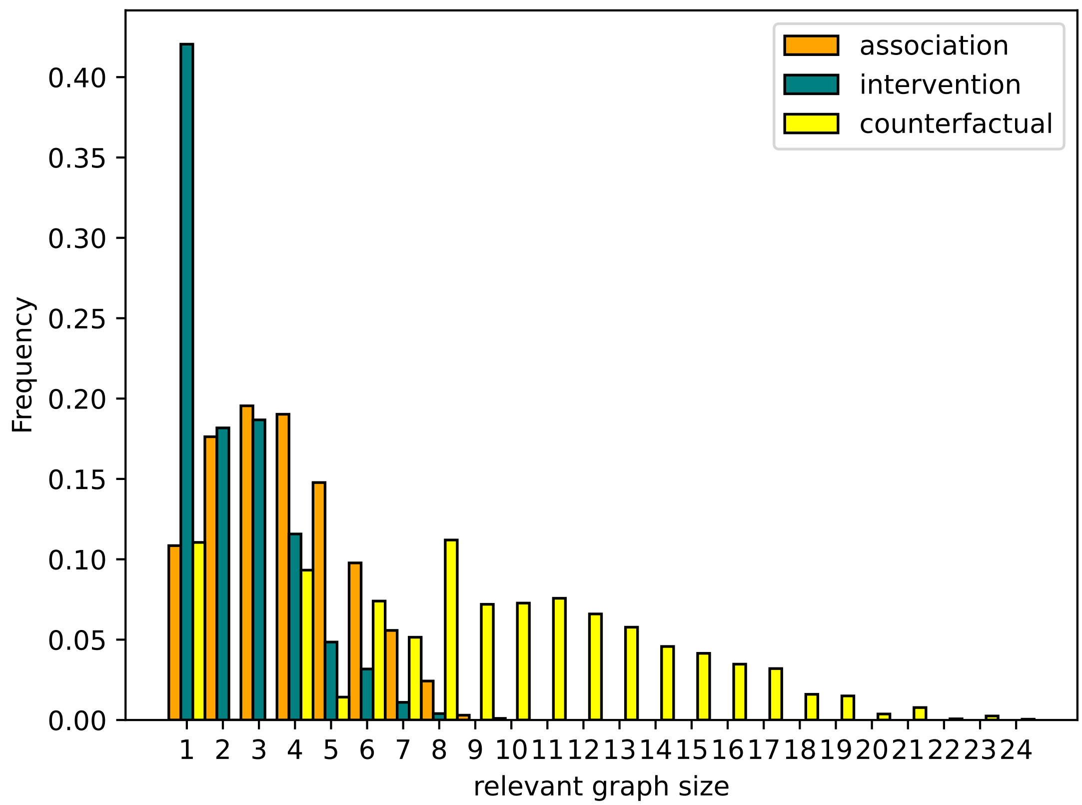

Figure 7: Distribution of query difficulty within each level, as measured by size of relevant subgraph defined in section 3.

A.1 Graph Modifications for Each Level

For an association level q of the form $p(\mathrm{v}_i \mid \mathrm{v}_j = v_j)$, we keep the same graph and query: $M' = M$ and $q' = q$.

For an intervention level query of the form $p(\mathrm{v}_{i(\mathrm{v}_j=c)})$, we modify the graph and query. We make $q' = p(\mathrm{v}_i)$, and M' a modification of M where we replace the mechanism for v_j with a constant function $f_j = c$, resulting with $\mathbb{F}' = \mathbb{F}_{-j} \cup \{f_j = c\}$. We also remove any incoming edges to v_j, resulting with $G' = (V, E \setminus \{(k \to j) \mid k \in V\})$.

For a counterfactual query $p(\mathrm{v}_{j(\mathrm{v}_i=c)} \mid \mathrm{v}_k = v_k)$, we modify the graph and query. We create M' by first augmenting it with a twin copy of each endogenous node v denoted $\mathrm{v}^{\mathrm{twin}}$. This results with $G' = (V \cup V^{\mathrm{twin}}, E \cup E^{\mathrm{twin}})$. E^{twin} mirrors E but connects twin copy nodes. The mechanisms for $\mathrm{v}_i^{\mathrm{twin}}$ uses parents in V^{twin} but share noise u_i with the original v_i. Then the same graph surgery and mechanism replacement is applied to the *twin* copy $\mathrm{v}_j^{\mathrm{twin}}$ to account for the intervention. We then define $q' = p(\mathrm{v}_j^{twin} \mid \mathrm{v}_k = v_k)$ in M'. See (Shpitser & Pearl, 2012) for more discussions on twin network graph that handles two possible worlds (enough for our setting) and its generalization *parallel worlds graph* that handles more.

All the modified queries q' are association level queries, so we can use standard graphical model inference, e.g. variable elimination Zhang & Poole (1994), in M' to compute the answer.

See fig. 2 for example graph modifications of each level.

A.2 Additional Dataset Details

The training set is sampled from 80 different graphical models, with 100 queries sampled per model. The development and test sets are sampled from 200 different graphical models, with 10/40 queries per model for dev/test.

B Additional Details of Experiment Setup

B.1 Additional Hyperparameters

RLVR For GRPO variant, we train 3B and 7B models for 7.5k steps, 32B models for 2.5k steps. For DAPO variant, we train 3B and 7B models for 2.5k steps, and 32B models for 850 steps. We use the final checkpoint for all. These step heuristically chosen to roughly control the amount of gpu hours spent on each run (32B is about 3x slower than 7B, and DAPO is roughly another 3x slower in terms of time per step due to filtering). For 3B models with GRPO and DAPO, the dev performance already plateaued early due to regression to direct prediction, so we did not train further beyond 7.5k/2.5k steps, respectively.

Additional hyper-parameters for our GRPO variant include a 0 coefficient on the KL term, a PPO clip ratio low of 0.2, high of 0.28, and c of 10, and token level averaging when computing the advantage function, following enhancements described in DAPO (aside from dynamic sampling) Yu et al. (2025b) and its implementation in VeRL (Sheng et al., 2024) . We use Adam optimizer (Kingma & Ba, 2017) with default parameters, weight decay of 0.1 and no warmup steps.

For DAPO runs, we focus on its curriculum aspect, disabling the overlong buffer, and setting a stricter filtering of accuracy between $(0.09, 0.91)$ which corresponds to more than 3 out of 32 rollouts being correct and more than 3 out of 32 rollouts being incorrect. We chose this filtering range in hopes of seeing stronger effects with filtering compared to $(0, 1)$ range originally used in DAPO (Yu et al., 2025b). However, our results showed no significant difference compared to GRPO.

C Prompts and Reasoning Outputs

System Prompts In fig. 8 we include the system prompt for RLVR and SFT. In fig. 9 we include the system prompt with hint for counterfactual level.

User Prompt In fig. 11, we show a full example of task input.

Reasoning Traces In figs. 12 to 15 we show reasoning traces of the four strategy categories “Incremental”, “Brute Force”, ‘Neighbors”, and “No Marginalization”.

D Additional Results

Accuracy Results In table 1 we show within-level generalization on filtered test set. In table 2 we show within-level generalization on unfiltered test set. In table 3 we show results for across-level generalization for filtered set. In table 4 we show results for across-level generalization for unfiltered set.

Hint Results In fig. 16 we show the performance of LLMs after RLVR with hint in system prompt (fig. 9) for the counterfactual level. It did not improve over the simple system prompt (fig. 8).

System Prompt RLVR

You are an expert on graphical models and causal inference. You task is to compute probability queries over a structural causal model.

Format your solution as follows:

THOUGHT PROCESS
All intermediate steps of analysis, reasoning, and computation.

ANSWER
Only state your final answer to the query, via a list of numbers such as [0.1, 0.7, 0.2] or [0.1, 0.9]. Round you answer to the nearest two digits.

System Prompt SFT

You are an expert on graphical models and causal inference. You task is to compute probability queries over a structural causal model.

Format your solution as follows:

Only state your final answer to the query, via a list of numbers such as [0.1, 0.7, 0.2] or [0.1, 0.9]. Round you answer to the nearest two digits.

Figure 8: System Prompt for RLVR and SFT.

System Prompt RLVR Hint For Counterfactual

You are an expert on graphical models and causal inference. Your task is to compute probability queries over a structural causal model (SCM).

Model Specification
You will be given the specification of an SCM:
* The graph structure will be provided.
* The parametrization of mechanisms will be specified with conditional probability tables (CPTs).

Each endogenous variable v_i is associated with a collection of independent exogenous variables-one for each possible joint assignment of its parents-that follows the same conditional distribution P(v_i | parents(v_i)). Given a joint assignment of its parents and the values of the corresponding exogenous variables, the mechanism assigns v_i the value of the appropriate exogenous variable.

Counterfactual Queries
To compute a counterfactual query:
1. Construct a twin network:
* Copy all endogenous variables.
* For each pair (v_i_original, v_i_copy), they share the same exogenous variables but connect to their respective parent sets (original vs. copied).
2. Interpret the query as a standard probability inference problem:
* Observations are applied to the original variables. * Interventions are applied to the copied variables. * The query target is also a copied variable.

Format
Format your solution as follows:
THOUGHT PROCESS
All intermediate steps of analysis, reasoning, and computation.

ANSWER
Only state your final answer to the query, via a list of numbers such as [0.1, 0.7, 0.2] or [0.1, 0.9]. Round you answer to the nearest two digits.

Figure 9: System Prompt for RLVR and SFT.

LLM Judge Results In table 5 we show numerical results for strategy categorization, visualized in fig. 5. In table 6 we show numerical results for strategy categorization on unfiltered test set,

LLM Prompt, Strategy Categorization

You are an expert grader that never makes mistakes.

Input
You will be given a LLM's SOLUTION for a QUESTION that asks for the marginal distribution of some random variable v_i under an intervention, observation, or counterfactual (hypothetical intervention under an observation).

Analysis
A correct strategy need to marginalize over other relevant variables correctly. You need to determine if the solution strategy is one of the following:
* immediate: attempt to marginalize only over immediate neighbors
* incremental: attempt to marginalize over neighbors as well as other more distant variables, performing
marginalization incrementally, often following the graph structure and performing summations locally over one subset of variables at a time for many times.
* brute: attempt to marginalize over neighbors as well as other more distant variables, but does so by
explicitly writing out a main formula that sums over the joint probability distribution over ALL relevant variables together (which often involves many terms), instead of summing over smaller subsets many times.
* none: no attempt at explicit marginalization.

Formatting Response
You need to output one judgement specified below: for that judgement, put any relevant EVIDENCE, which are excerpts from SOLUTION, within an <evidence></evidence> tag, then your EXPLANATION, if any, within the <explanation></explanation> tag, and finally put your judgement in <judgement></judgement> tag.
1. Overall Strategy
Use <evidence_strategy></evidence_strategy>, <explanation_strategy></explanation_strategy>, and
<judgement_strategy></judgement_strategy>. Choose judgement between "immediate", "incremental", "brute", "none".

LLM Prompt, Derivation Errors

You are an expert grader that never makes mistakes.
Input
You will be given a LLM's SOLUTION for a QUESTION that asks for the marginal distribution of some random variable v_i under an intervention, observation, or counterfactual (hypothetical intervention under an observation).

Analysis
A correct SOLUTION needs to perform derivations correctly and perform calculations correctly. Your task is to identify any probability derivation errors in the derivation. If you believe there are any errors the precise error location in the solution must be identified.

Probability derivation errors include:
* errors when applying probability identities (e.g. applying chain rule or bayes rule incorrectly, summing over too many or too few variables when marginalizing)
* false assumptions (e.g. ignoring dependencies between variables when performing inference, ignoring observations or interventions)
* ...

Probability derivation errors does NOT include:
* errors in copying CPT values
* numeric errors (e.g. incorrectly performing addition or multiplication)

Formatting Response
You need to output one judgement specified below: for that judgement, put any relevant EVIDENCE, which are excerpts from SOLUTION, within an <evidence></evidence> tag, then your EXPLANATION, if any, within the <explanation></explanation> tag, and finally put your judgement in <judgement></judgement> tag.

1. Derivation Error
Use <evidence_derivation_error></evidence_derivation_error>, <explanation_derivation_error></explanation_derivation_error>, and
<judgement_derivation_error></judgement_derivation_error>.

Choose your judgement from "yes" (solution contains derivations, and contains probability derivation errors), "no" (solution contains derivations, but no probability derivation errors detected), or "n/a" (solution does not contain any derivations).

Figure 10: Prompts for LLM Judge.

visualized in fig. 17. In table 7 we show numerical results for strategy categorization on the hard subset of the filtered test set, visualized in fig. 18.

Precision Results We plot precision of SFT vs. RL for all sizes and all levels in figs. 19 to 21.

	interv n10v2 easy (n=1086)	interv n10v2 medium (n=664)	interv n10v2 hard (n=348)	interv n10v2 all (n=2098)	assoc n10v2 easy (n=1684)	assoc n10v2 medium (n=1868)	assoc n10v2 hard (n=388)	assoc n10v2 all (n=3940)	counte n10v2 easy (n=24)	counte n10v2 medium (n=457)	counte n10v2 hard (n=231)	counte n10v2 all (n=712)
rl 32b	**100.000**	**99.398**	**85.632**	**97.426**	**96.793**	**68.201**	**43.299**	**77.970**	4.167	**17.068**	**16.450**	**16.433**
rl 32b curriculum	**100.000**	**99.096**	**84.195**	**97.092**	**95.843**	65.899	**39.433**	76.091	0.000	**16.849**	**16.883**	**16.292**
sft 32b	**99.724**	45.934	22.126	69.828	74.347	29.336	23.454	47.995	**29.167**	**19.037**	**15.152**	**18.118**
rl 7b	**99.908**	**91.566**	**58.621**	**90.419**	**81.354**	**40.685**	**33.505**	**57.360**	**4.167**	**13.348**	**15.152**	**13.624**
rl 7b curriculum	**99.724**	**91.566**	**54.023**	**89.561**	**80.523**	**40.096**	**35.309**	**56.904**	**0.000**	**14.004**	**14.719**	**13.764**
sft 7b	99.079	13.102	16.379	58.151	67.221	21.306	15.979	40.406	**20.833**	**14.880**	**15.584**	**15.309**
rl 3b	89.779	8.584	5.172	50.048	47.268	10.278	12.887	26.345	**4.167**	7.221	6.926	7.022
rl 3b curriculum	**97.145**	8.133	5.172	53.718	**62.589**	10.011	10.309	32.513	**0.000**	7.440	6.926	7.022
sft 3b	**97.698**	**12.349**	**11.782**	**56.435**	**63.420**	**15.150**	**17.268**	**35.990**	**0.000**	**10.284**	**12.554**	**10.674**

Table 1: Within level generalization (test, filtered). System accuracy (average CORRECT, see eq. (3)) when training and evaluating on queries from same level. Stratified by query level, and difficulty within each level, as measured by $|V_{\text{rel}}|$, the *size of the relevant subgraph* to the query variable. Note that difficulty is not comparable across different levels. The models are trained on a mix of small/medium/large questions. Systems not significantly worse than the best (with a monte-carlo paired permutation test with n=10000) are bolded.

	interv n10v2 easy (n=4661)	interv n10v2 medium (n=2428)	interv n10v2 hard (n=911)	interv n10v2 all (n=8000)	assoc n10v2 easy (n=3398)	assoc n10v2 medium (n=3831)	assoc n10v2 hard (n=771)	assoc n10v2 all (n=8000)	counte n10v2 easy (n=2440)	counte n10v2 medium (n=4533)	counte n10v2 hard (n=1027)	counte n10v2 all (n=8000)
rl 32b	**99.850**	**95.140**	**76.729**	**95.788**	**98.411**	**79.953**	**53.567**	**85.250**	**93.975**	**69.446**	**41.383**	**73.325**
rl 32b curriculum	99.592	93.781	**76.400**	95.188	97.793	77.839	48.768	83.513	**93.525**	**69.667**	**42.454**	**73.450**
sft 32b	93.177	37.068	26.894	68.600	84.255	47.612	36.187	62.075	86.926	50.849	29.893	59.162
rl 7b	**99.657**	**85.502**	**54.226**	**90.188**	**90.318**	**64.213**	**46.304**	**73.575**	**87.295**	61.196	**38.267**	**66.212**
rl 7b curriculum	**99.378**	**83.979**	**51.811**	89.287	**89.670**	**64.161**	**48.898**	**73.525**	**86.762**	**62.453**	**38.462**	**66.787**
sft 7b	85.518	18.328	22.722	57.975	77.340	37.745	26.070	53.438	80.492	41.562	22.590	51.000
rl 3b	76.958	7.661	6.806	47.938	63.420	22.971	17.510	39.625	66.025	29.385	11.490	38.263
rl 3b curriculum	78.502	7.372	6.806	48.750	71.307	22.240	14.656	42.350	74.877	28.877	11.977	40.737
sft 3b	**79.682**	**11.656**	**14.929**	**51.663**	**72.484**	**29.313**	**23.217**	**47.062**	**77.664**	**32.274**	**18.306**	**44.325**

Table 2: Within level generalization (test, unfiltered). System accuracy (average CORRECT, see eq. (3)) when training and evaluating on queries from same level. Stratified by query level, and difficulty within each level, as measured by $|V_{\text{rel}}|$, the *size of the relevant subgraph* to the query variable. Note that difficulty is not comparable across different levels. The models are trained on a mix of small/medium/large questions. Systems not significantly worse than the best (with a monte-carlo paired permutation test with n=10000) are bolded.

	interv n10v2 easy (n=1086)	interv n10v2 medium (n=664)	interv n10v2 hard (n=348)	interv n10v2 all (n=2098)
rl 32b asso	**100.000**	**99.398**	**85.920**	**97.474**
rl 32b coun	**100.000**	**98.946**	80.172	96.378
32b rl init	**99.908**	91.867	52.011	89.418
sft 32b asso	**99.724**	41.717	21.552	68.398
sft 32b coun	**99.908**	44.428	26.437	70.162
32b sft init	95.580	12.199	12.356	55.386

	assoc n10v2 easy (n=1684)	assoc n10v2 medium (n=1868)	assoc n10v2 hard (n=388)	assoc n10v2 all (n=3940)
rl 32b inte	**95.487**	**62.152**	**38.660**	**74.086**
rl 32b coun	**96.378**	**62.152**	**34.794**	**74.086**
32b rl init	88.539	39.186	22.680	58.655
sft 32b inte	72.031	26.927	24.485	45.964
sft 32b coun	57.423	26.927	22.938	39.569
32b sft init	58.314	11.991	10.825	31.675

	counte n10v2 easy (n=24)	counte n10v2 medium (n=457)	counte n10v2 hard (n=231)	counte n10v2 all (n=712)
rl 32b inte	**0.000**	**17.287**	**16.883**	**16.573**
rl 32b asso	**4.167**	**17.943**	**19.913**	**18.118**
32b rl init	**0.000**	12.473	9.091	10.955
sft 32b inte	**8.333**	**14.004**	**13.420**	13.624
sft 32b asso	**4.167**	11.160	12.554	11.376
32b sft init	**4.167**	8.315	7.359	7.865

	interv n10v2 easy (n=1086)	interv n10v2 medium (n=664)	interv n10v2 hard (n=348)	interv n10v2 all (n=2098)
rl 7b asso	**98.619**	**87.500**	**52.874**	**87.512**
rl 7b coun	**99.079**	**87.199**	44.540	**86.273**
7b rl init	61.510	51.958	16.092	50.953
sft 7b asso	**99.263**	19.729	18.391	60.677
sft 7b coun	97.514	18.825	22.126	60.105
7b sft init	24.862	4.217	0.287	14.252

	assoc n10v2 easy (n=1684)	assoc n10v2 medium (n=1868)	assoc n10v2 hard (n=388)	assoc n10v2 all (n=3940)
rl 7b inte	48.337	**34.904**	**25.258**	**39.695**
rl 7b coun	**55.641**	31.156	**23.454**	**40.863**
7b rl init	29.513	11.135	6.701	18.553
sft 7b inte	**56.116**	18.094	**21.907**	34.721
sft 7b coun	31.116	18.737	18.299	23.985
7b sft init	7.898	2.677	1.546	4.797

	counte n10v2 easy (n=24)	counte n10v2 medium (n=457)	counte n10v2 hard (n=231)	counte n10v2 all (n=712)
rl 7b inte	**0.000**	**13.567**	**11.688**	12.500
rl 7b asso	**4.167**	**14.880**	**15.584**	**14.747**
7b rl init	**4.167**	8.753	4.762	7.303
sft 7b inte	**8.333**	6.127	6.494	6.320
sft 7b asso	**8.333**	9.409	**9.957**	9.551
7b sft init	**4.167**	1.751	0.433	1.404

	interv n10v2 easy (n=1086)	interv n10v2 medium (n=664)	interv n10v2 hard (n=348)	interv n10v2 all (n=2098)
rl 3b asso	77.072	**9.337**	5.747	43.804
rl 3b coun	74.125	8.434	5.172	41.897
3b rl init	24.309	**10.392**	3.736	16.492
sft 3b asso	60.866	**10.994**	**12.931**	37.131
sft 3b coun	**93.370**	**11.446**	**14.080**	**54.290**
3b sft init	22.744	8.283	2.874	14.871

	assoc n10v2 easy (n=1684)	assoc n10v2 medium (n=1868)	assoc n10v2 hard (n=388)	assoc n10v2 all (n=3940)
rl 3b inte	**41.627**	11.456	11.340	24.340
rl 3b coun	35.926	9.636	9.794	20.888
3b rl init	9.561	5.621	4.897	7.234
sft 3b inte	38.717	**15.953**	**15.979**	**25.685**
sft 3b coun	16.983	**14.507**	**15.979**	15.711
3b sft init	14.014	8.458	10.567	11.041

	counte n10v2 easy (n=24)	counte n10v2 medium (n=457)	counte n10v2 hard (n=231)	counte n10v2 all (n=712)
rl 3b inte	**0.000**	5.689	**8.225**	6.320
rl 3b asso	**4.167**	6.565	**7.359**	6.742
3b rl init	**0.000**	4.595	2.165	3.652
sft 3b inte	**0.000**	**10.284**	**11.688**	**10.393**
sft 3b asso	**4.167**	**9.409**	**10.390**	**9.551**
3b sft init	**0.000**	4.814	3.030	4.073

Table 3: Across-level generalization (test, filtered). Row specify which level trained on, column specify which level evaluated on. System accuracy (average CORRECT, see eq. (3)) on evaluation sets of different difficulties, as measured by $|V_{\text{rel}}|$, the *size of the relevant subgraph* to the query variable. Note that difficulty is not comparable across different levels. The models are trained on a mix of easy/medium/hard questions. Systems not significantly worse than the best (with a monte-carlo paired permutation test with n=10000) are bolded.

	interv n10v2 easy (n=4661)	interv n10v2 medium (n=2428)	interv n10v2 hard (n=911)	interv n10v2 all (n=8000)
rl 32b asso	**98.048**	**93.287**	**77.827**	**94.300**
rl 32b coun	97.147	91.269	72.338	92.537
32b rl init	95.559	77.636	44.566	84.312
sft 32b asso	87.170	36.656	27.442	65.038
sft 32b coun	83.201	39.621	31.065	64.038
32b sft init	76.636	11.038	14.490	49.650

	assoc n10v2 easy (n=3398)	assoc n10v2 medium (n=3831)	assoc n10v2 hard (n=771)	assoc n10v2 all (n=8000)
rl 32b inte	**97.705**	**74.915**	**46.433**	**81.850**
rl 32b coun	**98.175**	**73.532**	41.634	80.925
32b rl init	93.320	51.031	24.903	66.475
sft 32b inte	83.196	45.784	35.279	60.662
sft 32b coun	75.544	46.150	34.112	57.475
32b sft init	67.039	23.727	16.083	41.388

	counte n10v2 easy (n=2440)	counte n10v2 medium (n=4533)	counte n10v2 hard (n=1027)	counte n10v2 all (n=8000)
rl 32b inte	92.131	68.564	**41.967**	72.338
rl 32b asso	**93.033**	**70.042**	**43.720**	**73.675**
32b rl init	88.074	55.306	28.140	61.812
sft 32b inte	79.426	48.577	28.627	55.425
sft 32b asso	76.967	48.687	29.309	54.825
32b sft init	64.590	31.370	14.411	39.325

	interv n10v2 easy (n=4661)	interv n10v2 medium (n=2428)	interv n10v2 hard (n=911)	interv n10v2 all (n=8000)
rl 7b asso	**98.777**	**79.572**	**46.652**	**87.013**
rl 7b coun	98.219	76.895	39.627	85.075
7b rl init	62.969	32.867	14.380	48.300
sft 7b asso	79.897	22.858	21.625	55.950
sft 7b coun	83.802	20.099	26.015	57.888
7b sft init	18.537	1.689	0.110	11.325

	assoc n10v2 easy (n=3398)	assoc n10v2 medium (n=3831)	assoc n10v2 hard (n=771)	assoc n10v2 all (n=8000)
rl 7b inte	72.543	**54.346**	**33.982**	**60.113**
rl 7b coun	**75.044**	49.987	**30.999**	58.800
7b rl init	37.905	14.748	7.134	23.850
sft 7b inte	70.983	34.299	29.053	49.375
sft 7b coun	58.711	35.787	29.053	44.875
7b sft init	12.919	3.211	1.686	7.187

	counte n10v2 easy (n=2440)	counte n10v2 medium (n=4533)	counte n10v2 hard (n=1027)	counte n10v2 all (n=8000)
rl 7b inte	**84.672**	**62.850**	**34.761**	**65.900**
rl 7b asso	**83.934**	60.379	**34.664**	64.263
7b rl init	60.615	31.568	12.561	37.988
sft 7b inte	73.361	35.892	19.182	45.175
sft 7b asso	65.533	40.393	19.669	45.400
7b sft init	31.148	9.618	1.753	15.175

	interv n10v2 easy (n=4661)	interv n10v2 medium (n=2428)	interv n10v2 hard (n=911)	interv n10v2 all (n=8000)
rl 3b asso	73.868	7.784	6.476	46.137
rl 3b coun	73.203	6.755	6.257	45.413
3b rl init	35.250	6.260	4.281	22.925
sft 3b asso	71.680	**12.891**	**17.124**	47.625
sft 3b coun	**79.446**	**12.191**	13.941	**51.575**
3b sft init	42.502	5.890	5.488	27.175

	assoc n10v2 easy (n=3398)	assoc n10v2 medium (n=3831)	assoc n10v2 hard (n=771)	assoc n10v2 all (n=8000)
rl 3b inte	**60.153**	23.388	17.121	38.400
rl 3b coun	57.004	21.874	15.045	36.138
3b rl init	26.574	10.232	7.393	16.900
sft 3b inte	**60.212**	**28.974**	**24.125**	**41.775**
sft 3b coun	49.205	**27.982**	**21.401**	36.362
3b sft init	35.227	16.967	12.192	24.262

	counte n10v2 easy (n=2440)	counte n10v2 medium (n=4533)	counte n10v2 hard (n=1027)	counte n10v2 all (n=8000)
rl 3b inte	**62.459**	29.164	12.658	37.200
rl 3b asso	60.615	29.274	12.074	36.625
3b rl init	28.443	15.420	6.134	18.200
sft 3b inte	**62.459**	**31.348**	**15.774**	**38.838**
sft 3b asso	59.713	**32.120**	**16.456**	**38.525**
3b sft init	20.902	10.170	3.603	12.600

Table 4: Across-level generalization (test, unfiltered). Row specify which level trained on, column specify which level evaluated on. System accuracy (average CORRECT, see eq. (3)) on evaluation sets of different difficulties, as measured by $|V_{\text{rel}}|$, the *size of the relevant subgraph* to the query variable. Note that difficulty is not comparable across different levels. The models are trained on a mix of easy/medium/hard questions. Systems not significantly worse than the best (with a monte-carlo paired permutation test with n=10000) are bolded.

	reasoning incremental (n=80)	reasoning brute (n=80)	reasoning immediate (n=80)	reasoning none (n=80)
rl 3b	0.000	0.000	20.000	80.000
rl 3b init	8.750	3.750	51.250	36.250
rl 7b	32.500	0.000	35.000	32.500
rl 7b init	25.000	1.250	46.250	27.500
rl 32b	33.750	0.000	36.250	30.000
rl 32b init	28.750	5.000	33.750	32.500

	reasoning incremental (n=80)	reasoning brute (n=80)	reasoning immediate (n=80)	reasoning none (n=80)
rl 3b	0.000	0.000	17.500	82.500
rl 3b init	6.250	8.750	66.250	18.750
rl 7b	32.500	0.000	37.500	30.000
rl 7b init	12.500	18.750	67.500	1.250
rl 32b	46.250	3.750	28.750	21.250
rl 32b init	22.500	17.500	38.750	21.250

	reasoning incremental (n=80)	reasoning brute (n=80)	reasoning immediate (n=80)	reasoning none (n=80)
rl 3b	0.000	0.000	17.500	82.500
rl 3b init	7.500	6.250	46.250	40.000
rl 7b	41.250	2.500	47.500	8.750
rl 7b init	26.250	5.000	55.000	13.750
rl 32b	45.000	8.750	38.750	7.500
rl 32b init	32.500	11.250	43.750	11.250

	derivation error yes (n=80)	derivation error no (n=80)	derivation error na (n=80)
rl 3b	41.250	31.250	27.500
rl 3b init	80.000	17.500	2.500
rl 7b	16.250	83.750	0.000
rl 7b init	40.000	57.500	2.500
rl 32b	5.000	95.000	0.000
rl 32b init	5.000	95.000	0.000

	derivation error yes (n=80)	derivation error no (n=80)	derivation error na (n=80)
rl 3b	46.250	23.750	28.750
rl 3b init	85.000	10.000	5.000
rl 7b	50.000	50.000	0.000
rl 7b init	67.500	32.500	0.000
rl 32b	27.500	71.250	1.250
rl 32b init	28.750	68.750	2.500

	derivation error yes (n=80)	derivation error no (n=80)	derivation error na (n=80)
rl 3b	78.750	6.250	13.750
rl 3b init	92.500	2.500	5.000
rl 7b	88.750	11.250	0.000
rl 7b init	93.750	6.250	0.000
rl 32b	77.500	22.500	0.000
rl 32b init	83.750	15.000	1.250

Table 5: LLM Judge Numerical Results on Filtered Set. See fig. 5 for visualization.

	reasoning incremental (n=80)	reasoning brute (n=80)	reasoning immediate (n=80)	reasoning none (n=80)
rl 3b	0.000	0.000	12.500	87.500
rl 3b init	10.000	5.000	51.250	33.750
rl 7b	25.000	3.750	36.250	35.000
rl 7b init	20.000	7.500	46.250	25.000
rl 32b	27.500	1.250	41.250	28.750
rl 32b init	26.250	1.250	40.000	32.500

	reasoning incremental (n=80)	reasoning brute (n=80)	reasoning immediate (n=80)	reasoning none (n=80)
rl 3b	0.000	0.000	7.500	92.500
rl 3b init	11.250	7.500	52.500	28.750
rl 7b	28.750	0.000	37.500	33.750
rl 7b init	20.000	23.750	46.250	10.000
rl 32b	36.250	2.500	31.250	30.000
rl 32b init	30.000	11.250	27.500	31.250

	reasoning incremental (n=80)	reasoning brute (n=80)	reasoning immediate (n=80)	reasoning none (n=80)
rl 3b	0.000	0.000	10.000	90.000
rl 3b init	6.250	0.000	46.250	47.500
rl 7b	18.750	0.000	40.000	41.250
rl 7b init	17.500	6.250	36.250	40.000
rl 32b	31.250	5.000	33.750	30.000
rl 32b init	18.750	10.000	25.000	46.250

	derivation error yes (n=80)	derivation error no (n=80)	derivation error na (n=80)
rl 3b	41.250	31.250	27.500
rl 3b init	80.000	17.500	2.500
rl 7b	16.250	83.750	0.000
rl 7b init	40.000	57.500	2.500
rl 32b	5.000	95.000	0.000
rl 32b init	5.000	95.000	0.000

	derivation error yes (n=80)	derivation error no (n=80)	derivation error na (n=80)
rl 3b	46.250	23.750	28.750
rl 3b init	85.000	10.000	5.000
rl 7b	50.000	50.000	0.000
rl 7b init	67.500	32.500	0.000
rl 32b	27.500	71.250	1.250
rl 32b init	28.750	68.750	2.500

	derivation error yes (n=80)	derivation error no (n=80)	derivation error na (n=80)
rl 3b	78.750	6.250	13.750
rl 3b init	92.500	2.500	5.000
rl 7b	88.750	11.250	0.000
rl 7b init	93.750	6.250	0.000
rl 32b	77.500	22.500	0.000
rl 32b init	83.750	15.000	1.250

Table 6: LLM Judge Numerical Results on Unfiltered Set. See fig. 17 for visualization.

User Prompt All Levels

Here's a structural causal model over discrete random variables. The Variables are v0, v1, v2, v3, v4, v5, v6, v7, v8, v9. Here are the Values they can take on.

v0 can take values in [0, 1]
v1 can take values in [0, 1]
v2 can take values in [0, 1]
v3 can take values in [0, 1]
v4 can take values in [0, 1]
v5 can take values in [0, 1]
v6 can take values in [0, 1]
v7 can take values in [0, 1]
v8 can take values in [0, 1]
v9 can take values in [0, 1]

Here's the causal directed acyclic graph (DAG):
strict digraph {
v0; v1; v2; v3; v4; v5; v6; v7; v8; v9; v3 → v0; v3 → v5; v4 → v1; v4 → v3; v4 → v7; v4 → v8; v4 → v9; v7 → v6; v8 → v2; v8 → v6; v8 → v7; v9 → v3; }

Here are the causal conditional probability tables (CPT) associated with the DAG:
CPTs for v4:
P(v4) = [0.51, 0.49]

CPTs for v8:
P(v8| v4=0) = [0.02, 0.98]
P(v8| v4=1) = [0.36, 0.64]

CPTs for v7:
P(v7| v8=0,v4=0) = [0.94, 0.06]
P(v7| v8=0,v4=1) = [0.25, 0.75]
P(v7| v8=1,v4=0) = [0.49, 0.51]
P(v7| v8=1,v4=1) = [0.58, 0.42]

CPTs for v2:
P(v2| v8=0) = [0.11, 0.89]
P(v2| v8=1) = [0.97, 0.03]

CPTs for v9:
P(v9| v4=0) = [0.95, 0.05]
P(v9| v4=1) = [0.42, 0.58]

CPTs for v3:
P(v3| v9=0,v4=0) = [0.46, 0.54]
P(v3| v9=0,v4=1) = [0.61, 0.39]
P(v3| v9=1,v4=0) = [0.7, 0.3]
P(v3| v9=1,v4=1) = [0.77, 0.23]

CPTs for v6:
P(v6| v7=0,v8=0) = [0.1, 0.9]
P(v6| v7=0,v8=1) = [0.5, 0.5]
P(v6| v7=1,v8=0) = [0.47, 0.53]
P(v6| v7=1,v8=1) = [0.53, 0.47]

CPTs for v5:
P(v5| v3=0) = [0.38, 0.62]
P(v5| v3=1) = [0.66, 0.34]

CPTs for v0:
P(v0| v3=0) = [0.96, 0.04]
P(v0| v3=1) = [0.86, 0.14]

CPTs for v1:
P(v1| v4=0) = [0.59, 0.41]
P(v1| v4=1) = [0.7, 0.3]

(**Included Only For Counterfactual level**: Furthermore, each variable v is assumed to depend deterministically on its parents pa(v) and a collection of independent exogenous selector variables, one for each possible joint assignment to pa(v), whose marginal distribution is defined to be p(v | pa(v)). Given a particular assignment to pa(v), v takes on the value of the selector variable corresponding to that particular assignment pa(v).)

Here's your Question: {Question}

Now start your solution process. Be precise.

Association Question
What is the marginal distribution of v6 iven it is observed that v8=1?

Intervention Question
What is the marginal distribution of v6 given we intervented to set v8 to 1?

Counterfactual Question

What is the marginal distribution of v3 given we first observed v6 = 0 and then intervened to set v8 to 1?

Figure 11: Example user prompt x_{user} containing causal graph and query.

Incremental Marginalization

THOUGHT PROCESS
Given the intervention do($v1 = 0$), the distribution of $v1$ is fixed at 0. This means we need to consider the conditional probability tables (CPTs) for the other variables given $v1 = 0$.
The CPT for $v6$ is given by:

$$P(v6|v0, v1 = 0) = \begin{cases} \\ P(v6|v0 = 0, v1 = 0) & \text{if } v0 = 0 \\ P(v6|v0 = 1, v1 = 0) & \text{if } v0 = 1 \end{cases}$$

From the CPTs provided:

$$P(v6|v0 = 0, v1 = 0) = [0.65, 0.35]$$

$$P(v6|v0 = 1, v1 = 0) = [0.04, 0.96]$$

Next, we need to find the marginal distribution of $v0$ given the intervention do($v1 = 0$). The intervention does not change the distribution of $v0$ directly, so we use the original CPT for $v0$.
The CPT for $v0$ is given by:

$$P(v0|v1 = 0, v5 = 0) = [0.61, 0.39]$$

$$P(v0|v1 = 0, v5 = 1) = [0.47, 0.53]$$

We need to consider the distribution of $v5$:

$$P(v5) = [0.24, 0.76]$$

Given $v1 = 0$, we need to find the marginal distribution of $v0$:

$$P(v0 = 0) = P(v0 = 0|v1 = 0, v5 = 0)P(v5 = 0) + P(v0 = 0|v1 = 0, v5 = 1)P(v5 = 1)$$

$$P(v0 = 0) = (0.61 \times 0.24) + (0.47 \times 0.76)$$

$$P(v0 = 0) = 0.1464 + 0.3572 = 0.5036$$

$$P(v0 = 1) = 1 - P(v0 = 0) = 1 - 0.5036 = 0.4964$$

Now, we can find the marginal distribution of $v6$:

$$P(v6 = 0) = P(v6 = 0|v0 = 0)P(v0 = 0) + P(v6 = 0|v0 = 1)P(v0 = 1)$$

$$P(v6 = 0) = (0.65 \times 0.5036) + (0.04 \times 0.4964)$$

$$P(v6 = 0) = 0.32734 + 0.019856 = 0.347196$$

$$P(v6 = 1) = 1 - P(v6 = 0) = 1 - 0.347196 = 0.652804$$

ANSWER
$[0.3472, 0.6528]$

Figure 12: Example trace with incremental marginalization. This example is from system `rl 32b init`, on a medium size intervention level query. Trace marginalizes first over v5, then over v0.

Brute Force Marginalization

THOUGHT PROCESS
To find the marginal distribution of $v1$ given $v9 = 0$, we need to compute $P(v1|v9 = 0)$. This can be done using the law of total probability and the given conditional probability tables (CPTs).
1. **Identify the relevant CPTs and variables:**
- $P(v1|v7, v9)$
- $P(v7|v6)$
- $P(v9|v0, v6)$
- $P(v6|v0)$
- $P(v0)$
2. **Express $P(v1|v9 = 0)$ using the law of total probability:**

$$P(v1|v9 = 0) = \sum_{v7,v6,v0} P(v1|v7, v9 = 0) \cdot P(v7|v6) \cdot P(v9 = 0|v0, v6) \cdot P(v6|v0) \cdot P(v0)$$

3. **Break down the computation:**
- Compute $P(v9 = 0|v0, v6)$ from the CPTs.
- Compute $P(v6|v0)$ from the CPTs.
- Compute $P(v0)$ from the CPTs.
- Compute $P(v7|v6)$ from the CPTs.
- Compute $P(v1|v7, v9 = 0)$ from the CPTs.
4. **Compute each term:**
- $P(v9 = 0|v0, v6)$:

$$P(v9 = 0|v0 = 0, v6 = 0) = 0.95, P(v9 = 0|v0 = 0, v6 = 1) = 0.6,$$

$$P(v9 = 0|v0 = 1, v6 = 0) = 0.31, P(v9 = 0|v0 = 1, v6 = 1) = 0.32$$

- $P(v6|v0)$:

$$P(v6 = 0|v0 = 0) = 0.32, \quad P(v6 = 1|v0 = 0) = 0.68, \quad P(v6 = 0|v0 = 1) = 0.03, \quad P(v6 = 1|v0 = 1) = 0.97$$

- $P(v0)$:

$$P(v0 = 0) = 0.65, \quad P(v0 = 1) = 0.35$$

- $P(v7|v6)$:

$$P(v7 = 0|v6 = 0) = 0.12, \quad P(v7 = 1|v6 = 0) = 0.88, \quad P(v7 = 0|v6 = 1) = 0.71, \quad P(v7 = 1|v6 = 1) = 0.29$$

- $P(v1|v7, v9 = 0)$:

$$P(v1 = 0|v7 = 0, v9 = 0) = 0.2, \quad P(v1 = 1|v7 = 0, v9 = 0) = 0.8,$$

$$P(v1 = 0|v7 = 1, v9 = 0) = 0.03, \quad P(v1 = 1|v7 = 1, v9 = 0) = 0.97$$

5. **Combine all terms:**

$$P(v1 = 0|v9 = 0) = \sum_{v7,v6,v0} P(v1 = 0|v7, v9 = 0) \cdot P(v7|v6) \cdot P(v9 = 0|v0, v6) \cdot P(v6|v0) \cdot P(v0)$$

$$P(v1 = 1|v9 = 0) = \sum_{v7,v6,v0} P(v1 = 1|v7, v9 = 0) \cdot P(v7|v6) \cdot P(v9 = 0|v0, v6) \cdot P(v6|v0) \cdot P(v0)$$

6. **Calculate the sums:**

$$P(v1 = 0|v9 = 0) = (0.2 \cdot 0.12 \cdot 0.95 \cdot 0.32 \cdot 0.65 + 0.2 \cdot 0.88 \cdot 0.6 \cdot 0.68 \cdot 0.65 + 0.03 \cdot 0.12 \cdot 0.31 \cdot 0.03 \cdot 0.35 + 0.03 \cdot 0.88 \cdot 0.32 \cdot 0.97 \cdot 0.35)$$

$$P(v1 = 1|v9 = 0) = (0.8 \cdot 0.12 \cdot 0.95 \cdot 0.32 \cdot 0.65 + 0.8 \cdot 0.88 \cdot 0.6 \cdot 0.68 \cdot 0.65 + 0.97 \cdot 0.12 \cdot 0.31 \cdot 0.03 \cdot 0.35 + 0.97 \cdot 0.88 \cdot 0.32 \cdot 0.97 \cdot 0.35)$$

7. **Perform the calculations:**

$$P(v1 = 0|v9 = 0) = 0.002496 + 0.030912 + 0.0000324 + 0.0078528 = 0.0412932$$

$$P(v1 = 1|v9 = 0) = 0.019968 + 0.35808 + 0.0003816 + 0.285632 = 0.6640622$$

8. **Round to two decimal places:**

$$P(v1 = 0|v9 = 0) \approx 0.04, \quad P(v1 = 1|v9 = 0) \approx 0.66$$

ANSWER
$[0.04, 0.66]$

Figure 13: Example Trace with brute force marginalization. This example is from system `rl 7b init`, on a medium size association level query.

Neighbor Marginalization

THOUGHT PROCESS
1. **Intervention on $v2$:** We need to compute the marginal distribution of $v1$ after setting $v2 = 0$. This means we need to consider the distribution of $v1$ given $v2 = 0$ and sum over all possible values of $v5$ (since $v5$ is a parent of $v1$).
2. **CPT for $v1$ given $v2 = 0$:** The CPT for $v1$ given $v2$ and $v5$ is:
- $P(v1|v2 = 0, v5 = 0) = [0.83, 0.17]$
- $P(v1|v2 = 0, v5 = 1) = [0.22, 0.78]$
3. **Marginalize over $v5$:** We need to sum over the probabilities of $v5$ (which is $P(v5) = [0.49, 0.51]$).
4. **Compute the marginal distribution:**
- $P(v1 = 0|v2 = 0) = P(v1 = 0|v2 = 0, v5 = 0) \cdot P(v5 = 0) + P(v1 = 0|v2 = 0, v5 = 1) \cdot P(v5 = 1)$
- $P(v1 = 1|v2 = 0) = P(v1 = 1|v2 = 0, v5 = 0) \cdot P(v5 = 0) + P(v1 = 1|v2 = 0, v5 = 1) \cdot P(v5 = 1)$
Let's compute these values step by step.

$$P(v1 = 0|v2 = 0) = 0.83 \cdot 0.49 + 0.22 \cdot 0.51 = 0.4067 + 0.1112 = 0.5179$$

$$P(v1 = 1|v2 = 0) = 0.17 \cdot 0.49 + 0.78 \cdot 0.51 = 0.0833 + 0.3978 = 0.4811$$

Rounding to two decimal places, we get:

$$P(v1 = 0|v2 = 0) = 0.52$$

$$P(v1 = 1|v2 = 0) = 0.48$$

ANSWER
$[0.52, 0.48]$

Figure 14: Example trace with summation over immediate parents. This example is from system `rl 7b init`, on a medium size intervention level query. Trace is inconsistent, and contains an error about ignoring v4. These solutions introduce ambiguity between brute-force summation and incremental summation since they are the trivial cases of both. Thus, we treat them as a separate category. The system

No Marginalization

THOUGHT PROCESS
The question asks for the marginal distribution of v7 given that v4=1. This means we need to compute the probability distribution of v7 without conditioning on v4, but we can use the conditional probability of v4 given v5 to find the marginal distribution of v7.
ANSWER
$[0.31, 0.69]$

Figure 15: Example trace with no explicit marginalization. This example is from system `rl 3b`, on a small size association level query. Trace is inconsistent, and contains an error about ignoring v4.

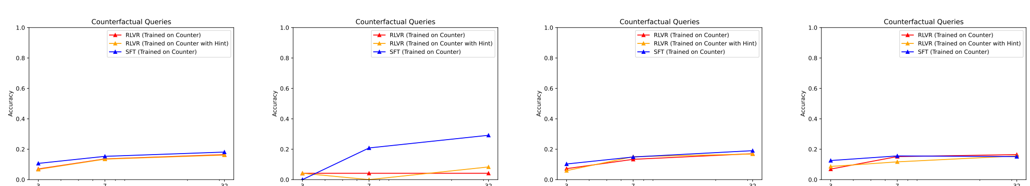

Figure 16: Counterfactual level with hint. Prompting with a hint about how to solve counterfactual queries by twin-network-graph is not enough to induce genuine solutions and improve performance post-RLVR. From left to right is accuracy breakdown on **all**, **small**, **medium**, and finally **large** problems. Having hint in the prompt did not significantly improve RLVR's performance on counterfactual level.

	reasoning incremental (n=80)	reasoning brute (n=80)	reasoning immediate (n=80)	reasoning none (n=80)
rl 3b	0.000	0.000	3.750	96.250
rl 3b init	11.250	12.500	48.750	27.500
rl 7b	97.500	0.000	2.500	0.000
rl 7b init	40.000	6.250	52.500	1.250
rl 32b	93.750	5.000	1.250	0.000
rl 32b init	82.500	15.000	1.250	1.250

	reasoning incremental (n=80)	reasoning brute (n=80)	reasoning immediate (n=80)	reasoning none (n=80)
rl 3b	0.000	0.000	6.250	93.750
rl 3b init	11.250	10.000	47.500	31.250
rl 7b	67.500	1.250	20.000	11.250
rl 7b init	11.250	62.500	23.750	2.500
rl 32b	65.000	27.500	5.000	2.500
rl 32b init	26.250	56.250	12.500	5.000

	reasoning incremental (n=80)	reasoning brute (n=80)	reasoning immediate (n=80)	reasoning none (n=80)
rl 3b	0.000	0.000	8.750	91.250
rl 3b init	13.750	2.500	28.750	55.000
rl 7b	57.500	3.750	35.000	3.750
rl 7b init	23.750	13.750	50.000	12.500
rl 32b	71.250	7.500	18.750	2.500
rl 32b init	45.000	17.500	22.500	15.000

	derivation error yes (n=80)	derivation error no (n=80)	derivation error na (n=80)
rl 3b	41.250	31.250	27.500
rl 3b init	80.000	17.500	2.500
rl 7b	16.250	83.750	0.000
rl 7b init	40.000	57.500	2.500
rl 32b	5.000	95.000	0.000
rl 32b init	5.000	95.000	0.000

	derivation error yes (n=80)	derivation error no (n=80)	derivation error na (n=80)
rl 3b	46.250	23.750	28.750
rl 3b init	85.000	10.000	5.000
rl 7b	50.000	50.000	0.000
rl 7b init	67.500	32.500	0.000
rl 32b	27.500	71.250	1.250
rl 32b init	28.750	68.750	2.500

	derivation error yes (n=80)	derivation error no (n=80)	derivation error na (n=80)
rl 3b	78.750	6.250	13.750
rl 3b init	92.500	2.500	5.000
rl 7b	88.750	11.250	0.000
rl 7b init	93.750	6.250	0.000
rl 32b	77.500	22.500	0.000
rl 32b init	83.750	15.000	1.250

Table 7: LLM Judge Numerical Results on Large Complexity Split of Filtered Test Set. See fig. 18 for visualization.

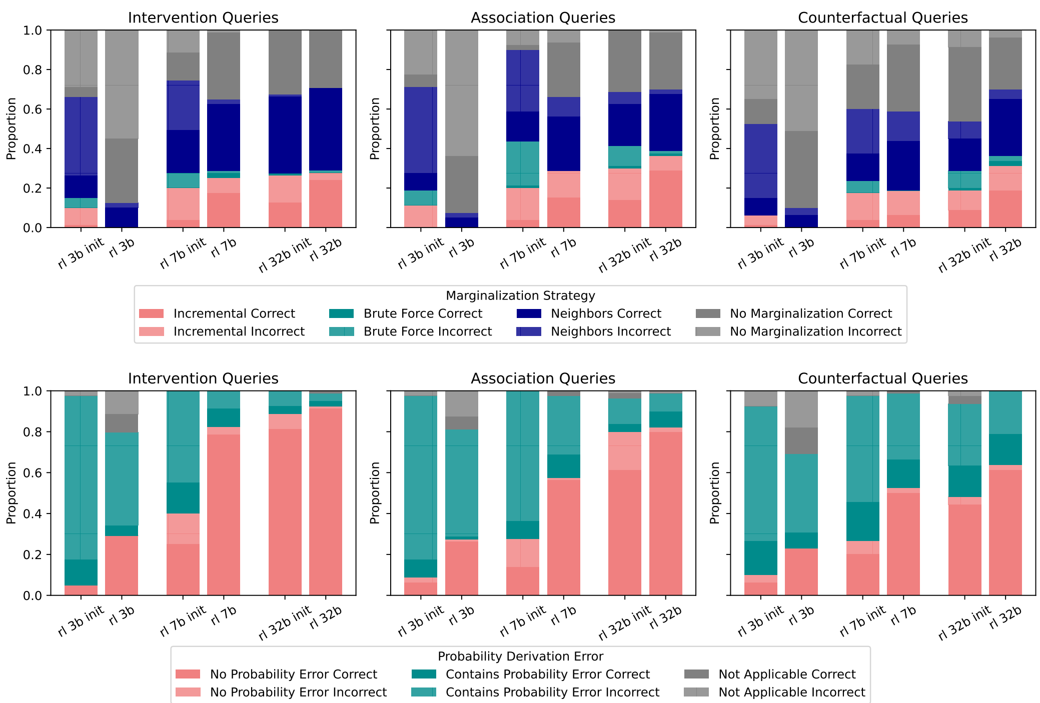

Figure 17: LLM judge analysis of reasoning strategy (top) and existence of derivation errors (bottom) before and after RLVR on **unfiltered** test set. Marginalization strategies are annotated on 80 samples per level. Derivation errors are also annotated on the same 80 samples per level.

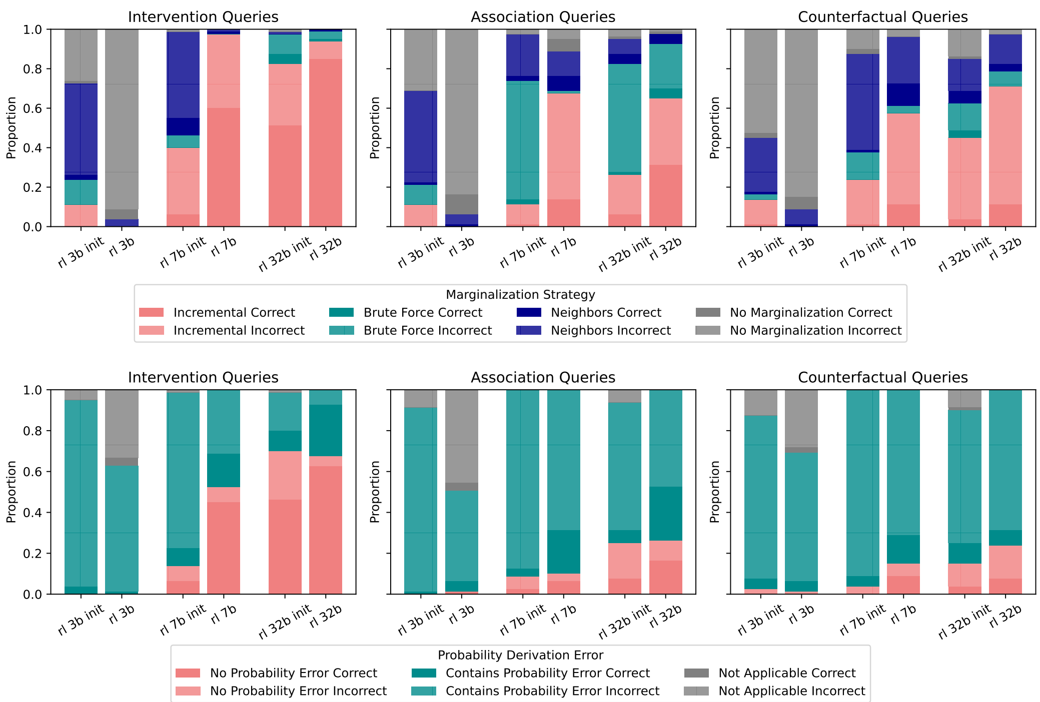

Figure 18: LLM judge analysis of reasoning strategy (top) and existence of derivation errors (bottom) before and after RLVR on the **large** complexity queries. Marginalization strategies are annotated on 80 samples per level. Derivation errors are also annotated on the same 80 samples per level.

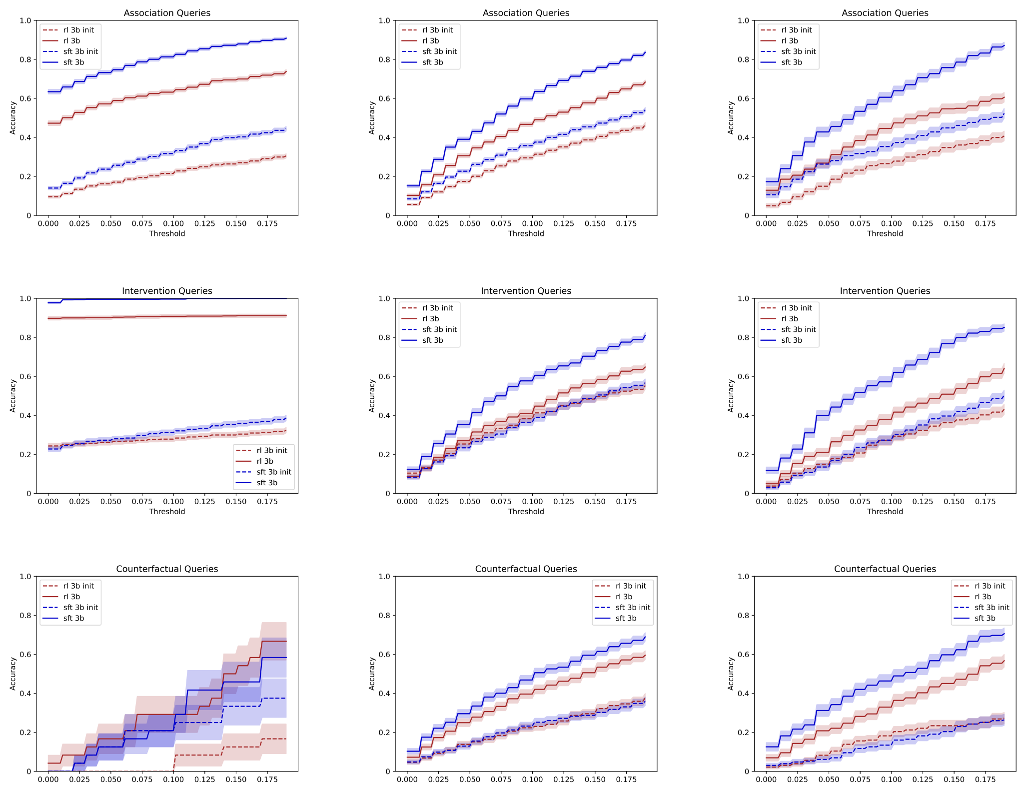

Figure 19: Accuracy by threshold $t \in (0, 0.2]$ for 3B Models. x-axis plots threshold for accuracy t (the lower the stricter). y-axis is accuracy. From top to bottom are different levels. From left to right are query complexities small, medium, large within each level.

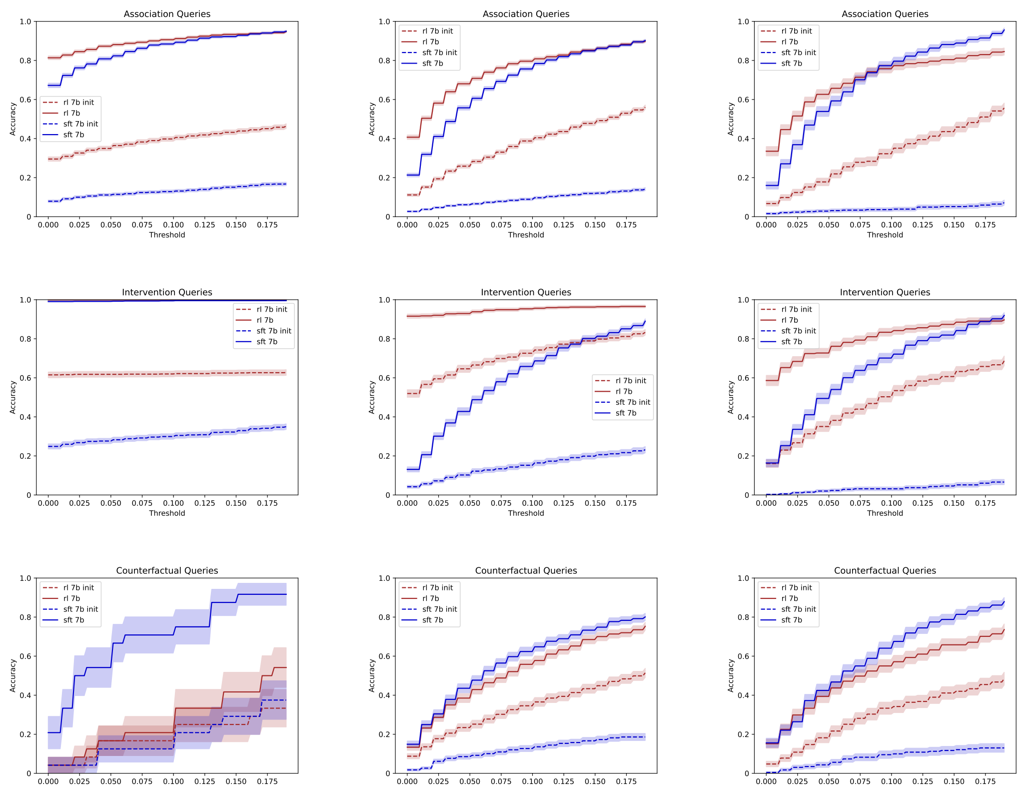

Figure 20: Accuracy by threshold $t \in (0, 0.2]$ for 7B Models. x-axis plots threshold for accuracy t (the lower the stricter). y-axis is accuracy. From top to bottom are different levels. From left to right are query complexities small, medium, large within each level.

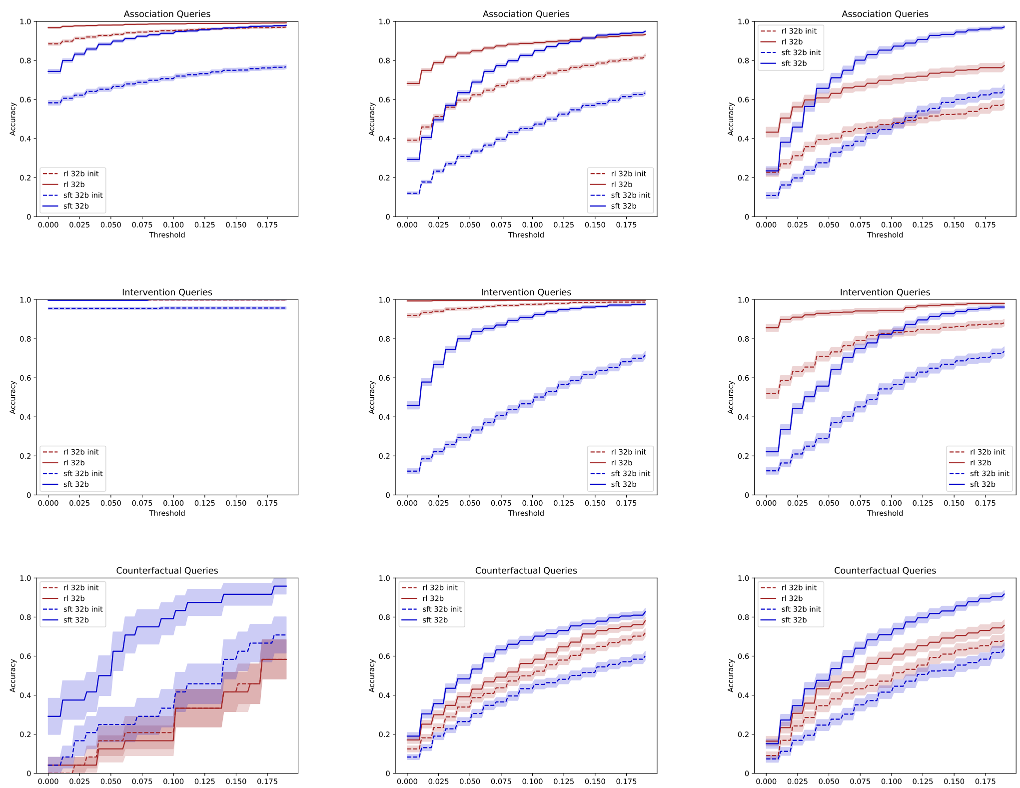

Figure 21: Accuracy by threshold $t \in (0, 0.2]$ for 32B Models. x-axis plots threshold for accuracy t (the lower the stricter). y-axis is accuracy. From top to bottom are different levels. From left to right are query complexities small, medium, large within each level.