BETTER THINK THRICE: LEARNING TO REASON CAUSALLY WITH DOUBLE COUNTERFACTUAL CONSISTENCY

Victoria Lin[1*] **Xinnuo Xu**[2] **Rachel Lawrence**[2] **Risa Ueno**[2] **Amit Sharma**[3]
Javier González[2] **Niranjani Prasad**[2]
[1]Carnegie Mellon University [2]Microsoft Research Cambridge [3]Microsoft Research India
vlin2@andrew.cmu.edu niranjani.prasad@microsoft.com

ABSTRACT

Despite their strong performance on reasoning benchmarks, large language models (LLMs) have proven brittle when presented with counterfactual questions, suggesting weaknesses in their causal reasoning ability. While recent work has demonstrated that labeled counterfactual tasks can be useful benchmarks of LLMs' causal reasoning, producing such data at the scale required to cover the vast potential space of counterfactuals is limited. In this work, we introduce *double counterfactual consistency* (DCC), a lightweight inference-time method for measuring and guiding the ability of LLMs to reason causally. Without requiring labeled counterfactual data, DCC verifies a model's ability to execute two important elements of causal reasoning: causal intervention and counterfactual prediction. Using DCC, we evaluate the causal reasoning abilities of various leading LLMs across a range of reasoning tasks and interventions. Moreover, we demonstrate the effectiveness of DCC as a training-free test-time rejection sampling criterion and show that it can directly improve performance on reasoning tasks across multiple model families.

1 INTRODUCTION

Modern large language models (LLMs) have achieved impressive performance on a wide range of language tasks, including tasks that require them to *reason* about their answers. Despite their strong performance on reasoning benchmarks, however, LLMs have shown themselves to be brittle when evaluated on data that deviates slightly in distribution from their training data. That is, there exists a gap in performance between tasks in which the LLM can rely on recall of statistical patterns—i.e., memorization—and tasks that require true reasoning ability Hüyük et al. (2025).

In particular, LLMs appear to exhibit difficulties with *causal reasoning*. When presented with *counterfactual* questions, for instance—questions in which some information is provided that potentially challenges the LLM's prior world state—models exhibit degraded performance compared to similar questions that do not contain counterfactual information. To better evaluate models' causal reasoning abilities, recent works have proposed benchmarks and evaluation strategies based on labeled counterfactual questions González & Nori (2024); Xu et al. (2025). However, this type of data is difficult to obtain at scale, often requiring extensive human intervention; and the potential space of counterfactuals is large, making exhaustive benchmarks difficult to construct.

In this paper, we address this gap with *double counterfactual consistency* (DCC), a lightweight inference-time method to evaluate and guide a model's ability to reason causally. DCC builds on a core intuition for the failure of LLMs to perform on causal reasoning tasks. Without requiring labeled counterfactual data, DCC verifies a model's ability to execute two important elements of causal reasoning: *causal intervention* and *counterfactual prediction*—which together suggest an understanding of the underlying causal graph.

Concretely, DCC guides a model to (1) answer a standard reasoning question; (2) apply an intervention to produce a counterfactual version of that question; and (3) apply a second intervention that inverts the

[*]Work done while author was an intern at Microsoft Research Cambridge.

previous intervention, yielding a "double counterfactual" question that should have an answer equal to the original. From this, we define the model's double counterfactual consistency: consistency between the answer to the original question and the answer to the double counterfactual question. This measure relies on the model's correct execution of both intervention steps, as well as the implicit correctness of the intermediate counterfactual. We posit that models that maintain double counterfactual consistency for a large proportion of answers are therefore more likely to be capable of robust causal reasoning than those that do not.

An important advantage of DCC is that it does not rely on the inherent correctness of the model: that is, a model may be wrong on the original question but still display double counterfactual consistency if its answers to the original and double counterfactual questions agree (and vice versa). This distinction allows DCC to disentangle two separate properties of the model: its base accuracy and its causal reasoning ability. This is critical in cases where strong base model performance may mask deficiencies in causal reasoning. For instance, a model with low base accuracy but strong causal reasoning and a model with high accuracy but weak causal reasoning may achieve similar accuracy on counterfactual questions, but only the former maintains robustness under intervention. By isolating consistency under intervention, DCC provides a direct signal of causal reasoning that is not confounded by surface-level accuracy.

In addition to serving as a measure of LLMs' causal reasoning ability, DCC can also be used as a practical method to improve model performance on causal reasoning tasks. In particular, DCC provides a natural criterion for inference-time rejection sampling. Given a question, if the model's answers to the original and double counterfactual prompts are inconsistent, the response can be rejected and the model resampled until double counterfactual consistency is achieved. This allows DCC to function not only as an evaluative tool but also as a lightweight mechanism for guiding inference.

In our experiments, we examine how models' DCC varies across datasets and types of interventions, providing insight into the scenarios where models are more or less capable of maintaining causal consistency. Interestingly, we find that DCC is not strictly correlated with a model's base performance: models with higher accuracy on factual reasoning tasks do not necessarily achieve higher DCC. This suggests that DCC captures a distinct property of model behavior, complementary to standard accuracy metrics. We further demonstrate that using DCC as a rejection sampling criterion can improve accuracy on counterfactual reasoning benchmarks compared to directly querying the model, as well as compared to common inference-time baselines such as in-context learning. These findings illustrate the dual role of DCC: it provides a principled metric for evaluating causal reasoning while also serving as a practical tool to guide model behavior.

2 WHY DO LLMS STRUGGLE TO ANSWER COUNTERFACTUAL QUESTIONS?

In this section, we discuss why LLMs struggle with causal reasoning, which is most clearly illustrated by their failure to perform on counterfactual questions. We begin by considering an idealized LLM that possesses true causal reasoning ability. Such a model should be able to reason coherently over any possible question a user could ask, including those involving *interventions* that alter the underlying world state. In practice, however, LLMs are only trained on a small portion of the full question space: the "factual" region corresponding to data observed during training (Wu et al., 2024).

For example, training data may contain factual questions such as:

Q: Is 6 divisible by 2? A: Yes

If we instead pose a counterfactual question with an intervention that changes the premise, such as:

Q: *Suppose 6 is a prime number.* Then is 6 divisible by 2?

the model encounters two distinct challenges. First, it must correctly interpret the intervention itself—understanding that the question now operates in a modified world state. Second, it must infer the consequence of this intervention, i.e., the counterfactual outcome under the new assumptions. In other words, the model must both recognize the shift in causal context and be able to reason coherently within it. A model that succeeds in both steps can be said to have internalized (at least behaviorally) the causal structure linking the question, the intervention, and the outcome.

$$T = 0, Z \rightarrow \widehat{Y}(T = 0, Z) \xrightarrow{\text{do}(T=1)} \widehat{Y}(T = 1, Z) \xrightarrow{\text{do}(T=0)} \widehat{Y}'(T = 0, Z)$$

Figure 1: Double counterfactual consistency.

Algorithm 1 Double counterfactual consistency

Require: Question $Q_f = (T = 0, Z)$; LLM f

1: Query model f with Q_f to obtain the initial answer $\widehat{Y}(T = 0, Z)$
2: Construct counterfactual question Q_{cf} by adding intervention $\text{do}(T = 1)$ to Q_f; query f to obtain counterfactual answer $\widehat{Y}(T = 1, Z)$
3: Construct double counterfactual question Q'_f by adding second intervention $\text{do}(T = 0)$ to Q_{cf}; query f again to obtain $\widehat{Y}'(T = 0, Z)$
4: Compute double counterfactual consistency as: $\text{DCC}(T, Z) = \mathbb{1}\{\widehat{Y}(0, Z) = \widehat{Y}'(0, Z)\}$

Formalization. We now link this intuition to the framework of interventions and counterfactuals in statistical causal inference. In causal inference, counterfactual outcomes are by definition unobservable. Given a binary intervention T (e.g., a medical treatment) and a set of covariates Z, only one outcome Y can be observed for a specific individual at a specific moment in time. Letting $Y(t)$ denote the potential outcome (Rubin, 2005), or the outcome that *would* have occurred under intervention $T = t$, we observe $Y = Y(T)$ while the counterfactual outcome $Y(1 - T)$ remains unobserved.

Extending this framework to reasoning tasks, we consider pairs of questions and answers (Q, Y). Let each question Q be parameterized as (T, Z), where T encodes the factual or counterfactual element of the question ($T = 0$ for factual, $T = 1$ for counterfactual), and Z represents the remaining content of the question. Let $Y(\cdot)$ denote the potential answer function over T and Z. An ideal model perfectly learns the full potential outcome space over all possible questions, i.e., $P(Y(T = t, Z = z))$ for all t, z, thereby capturing the causal relationship between T and Y.

In practice, however, models are trained only on observational factual data drawn from $P(Y \mid T = 0, Z)$, enabling them to approximate only the factual portion of the potential outcome space, $P(Y(T = 0, Z))$. Having never observed counterfactual examples from $P(Y \mid T = 1, Z)$, such models are unable to generalize to the counterfactual regime $P(Y(T = 1, Z))$, which is necessary for robust causal reasoning. Counterfactual datasets are therefore useful diagnostic tools for exposing failures in causal reasoning—but constructing them at scale remains difficult. While limited counterfactual data can be generated programmatically, the space of possible interventions is effectively unbounded, and existing approaches typically focus on narrow, domain-specific perturbations (Xu et al., 2025; Huang et al., 2025).

This motivates two central challenges:

1. How can we *evaluate* whether a model has learned $P(Y(T = 1, Z))$ without the need to construct new counterfactual datasets $P(Y \mid T = 1, Z)$ for every possible T?
2. How can a given model *learn* to approximate $P(Y(T = 1, Z))$ without direct access to $P(Y \mid T = 1, Z)$? Viewed differently, this represents a distribution shift from the factual ($T = 0$) to counterfactual ($T = 1$) domain.

3 Double Counterfactual Consistency

To address these challenges, we propose *double counterfactual consistency* (DCC). As detailed in Figure 1 and Algorithm 1, DCC evaluates whether a model's answers remain consistent after performing and then inverting a causal intervention. Specifically, it compares the model's prediction for the original question, $\widehat{Y}(T{=}0, Z)$, with its prediction for the double counterfactual version of the question, $\widehat{Y}'(T{=}0, Z)$, obtained by first applying and then reversing an intervention sequence $T = 0 \rightarrow \text{do}(T{=}1) \rightarrow \text{do}(T{=}0)$.

Because the second intervention should reverse the first, the two predictions should match. We define DCC as an indicator of this equivalence:

$$\text{DCC}(T, Z) = \mathbb{1}\{\widehat{Y}(T = 0, Z) = \widehat{Y}'(T = 0, Z)\}$$

Checking DCC softly checks the model's ability to execute two important elements of causal reasoning: intervention and counterfactual prediction. In particular, the model will find the two answers to be equivalent if it can (i) correctly apply causal interventions, as the model must not only intervene but also act to undo that intervention, and (ii) model the counterfactual outcome distribution, since the correctness of the double counterfactual depends on the correctness of the intermediate counterfactual.

We implement the double counterfactual consistency procedure with a prompt template that allows the model to complete all three DCC reasoning steps—factual, counterfactual, and double counterfactual—within a single generation trace (Listing 1). We induce the model to follow this format through in-context learning (ICL), providing a small number of examples constructed from labeled counterfactual data. This allows us to flexibly and scalably measure DCC for different datasets and interventions, since the procedure relies on only a handful of labeled examples in each case.

Importantly, DCC can be used not only to evaluate models' causal reasoning ability but also to improve it. We propose three complementary ways in which double counterfactual consistency can be applied: as a metric, as an inference-time criterion, and as a training-time reward.

Metric for causal reasoning. DCC can serve as a quantitative measure of an LLM's causal reasoning ability. Over a test dataset, the proportion of samples satisfying DCC provides a metric that enables comparison across models, datasets, or types of interventions. Unlike accuracy-based metrics, DCC evaluates whether a model maintains causal coherence under interventions, independent of the factual correctness of its answers.

Inference-time causal consistency verification. At inference time, DCC can be used directly as a criterion for rejection sampling to ensure that model outputs are causally consistent. For instance, a model may already be capable of intervention and counterfactual prediction, but perhaps is inconsistent in its generations, sometimes relying on memorization as a shortcut. To filter out responses where the model fails to reason causally, we prompt the model to follow the three-step DCC reasoning process and *sample until agreement*, i.e., sample generations until the first and last answer agree. While the multiple generations introduce some computational overhead, inference-time costs from rejection sampling remain far less computationally demanding than fine-tuning. Moreover, we find that in practice, agreement is reached fairly quickly: across all of our experiments, a mean of only 3.97 attempts is required to achieve agreement.

Post-training causal consistency objective. Finally, DCC can be incorporated as a reward in test-time training with reinforcement learning (Zuo et al., 2025), to encourage structured counterfactual reasoning behavior. Specifically, given a test batch of task prompts, we fine-tune parameter-efficient LoRA adapters (Hu et al., 2022) on this batch using group-robust policy optimization (GRPO) (Shao et al., 2024), with the DCC criterion as reward. By guiding the model through the DCC reasoning steps and rewarding instances where consistency holds, the model can be explicitly incentivized to learn both how to perform interventions and how to model counterfactual outcomes, reinforcing representations that capture underlying causal structure.

Implementing DCC as a single reasoning trace renders it highly scalable, lightweight, and easy to adopt: in this form, it can be flexibly used as a metric, rejection sampling criterion, or RL reward, all without complex infrastructure. It requires no specialized training setup and integrates easily with standard generation pipelines. However, we note that this implementation may also allow the model to learn certain shortcuts when DCC is used as a post-training reward. Because the model has simultaneous access to its predictions for both the original answer and the double counterfactual answer, it may learn simply that it is rewarded when the two answers are the same. We find that early stopping during training is key to preventing the model from overfitting and exploiting this shortcut.

Furthermore, for applications with greater sensitivity to shortcutting, it is also possible to generate the three DCC reasoning steps as separate traces, with each subsequent step receiving only the answer from the previous step. This eliminates direct access between the first and third steps. However, we view this as an implementation that should be used only when necessary given the increased computational overhead and the added complexity of backpropagating gradients through three independent model calls.

Listing 1 DCC prompt template

```
Question: [Original question]
<reasoning>[Reasoning trace]</reasoning>
<answer>[Original answer]</answer>

Now consider a counterfactual version of this question, where the following intervention is made:
[Intervention]
What is the answer?
<reasoning>[Reasoning trace]</reasoning>
<answer>[CF answer]</answer>

Now consider a counterfactual version of the *counterfactual question, where the following change
↪  intervenes on the previous intervention to restore the question to its original state:
[Second intervention]
What is the answer?
<reasoning>[Reasoning trace]</reasoning>
<answer>[Double CF answer]</answer>
```

4 EXPERIMENTS

In our experiments, we evaluate DCC both as a diagnostic measure of causal reasoning and as a mechanism for improving model performance. We first assess DCC as an evaluation metric across a range of established reasoning benchmarks, computing the proportion of samples for which consistency holds. We then examine whether DCC—either as an inference-time rejection sampling criterion or as a post-training reward—can help improve performance on causal reasoning tasks.

4.1 BENCHMARKS AND PERTURBATIONS

GSM8K. GSM8K (Cobbe et al., 2021) is a benchmark for multi-step mathematical reasoning consisting of grade school-level math word problems that require combining arithmetic and logical operations.

CruxEval. CruxEval (Gu et al., 2024) is a benchmark consisting of short Python functions, where the goal is to complete assert statements verifying the input or output of the function. CruxEval focuses on symbolic and procedural reasoning, testing whether models can internalize and apply programmatic logic.

MATH. MATH (Hendrycks et al., 2021) is a benchmark consisting of competition-level math problems spanning algebra, geometry, number theory, and other advanced topics. Compared to GSM8K, MATH problems require substantially deeper compositional reasoning and more formal symbolic manipulation.

The following counterfactual datasets are used only to evaluate DCC as a method for improving performance. Both datasets are created using RE-IMAGINE (Xu et al., 2025), a framework for programmatically generating counterfactual variants of reasoning problems. For datasets where problems can be symbolically represented (e.g., GSM8K), RE-IMAGINE applies controlled *mutations* that alter the problem in well-defined ways. Mutations are categorized by whether they preserve or modify the underlying logic.

RE-IMAGINE GSM8K. For GSM8K, we focus on one representative mutation of each type: `IrrelevantInfoHard`, which adds distracting information that is ultimately irrelevant to the answer (i.e., preserving the underlying logic), and `InsertConditional`, which introduces a conditional statement that changes the reasoning needed to reach the correct answer (i.e., changing the underlying logic).

RE-IMAGINE CruxEval. For CruxEval, we focus on a mutation that modifies the underlying logic of the question: `MutateStringCF`, which adds a counterfactual statement changing a string instance in the program to a randomly generated character sequence of the same length.

Together, these datasets form a progression of reasoning challenges, from standard benchmarks (GSM8K, CruxEval, MATH) to their systematically perturbed or counterfactual variants. The most direct tests of causal reasoning arise in the RE-IMAGINE datasets, where success requires reasoning under explicit counterfactual assumptions. While causal reasoning may also yield empirical benefits on standard benchmarks, such gains may be obscured because leading LLMs have largely memorized these datasets, and DCC may reduce their reliance on shortcut-based answers (to the detriment of their performance).

4.2 Models and Baselines

We evaluate and finetune a diverse mix of frontier and lightweight open source models (loaded via Unsloth with 4-bit quantization + LoRA): Llama 3.1 8B, Llama 3.2 3B, Phi-4, Phi-3.5 Mini, Qwen3 8B, and Qwen3 4B. Qwen models optionally use a `/nothink` suffix (flag) to disable explicit reasoning tokens.

Base model. The model is queried directly with questions from the benchmark. Each question elicits a single reasoning trace and answer. This baseline establishes the model's unmodified performance on the benchmark without any additional prompting or sampling strategies.

ICL. Two labeled in-context examples from the benchmark are included in the prompt, following the same query format as in the base model baseline. This setup controls for the impact of in-context demonstrations, allowing us to assess how much of DCC's performance can be attributed simply to exposure to example reasoning traces (which are necessary to induce the model to follow the reasoning format in in Listing 1) rather than to the DCC procedure itself.

5 Results and Discussion

We report our results evaluating DCC both as a metric for causal reasoning and as a method for improving performance on reasoning tasks. In Section 5.1, results are computed over the full test set of the reported benchmark, excluding 10 samples from which we draw in-context learning examples. In Section 5.2, results are computed as an average over 15 64-sample batches drawn at random from the reported benchmark, again excluding the 10 samples used as in-context learning examples.

5.1 DCC as a causal reasoning metric

To evaluate the usefulness of DCC as a causal reasoning metric—in particular whether explicitly testing a model's ability to intervene and predict counterfactuals provides information beyond standard accuracy—we compare it against agreed-answer accuracy, wherein the model is queried twice with the same question, and an answer is considered valid only if the two responses agree. Accuracy is then computed using these agreed answers.

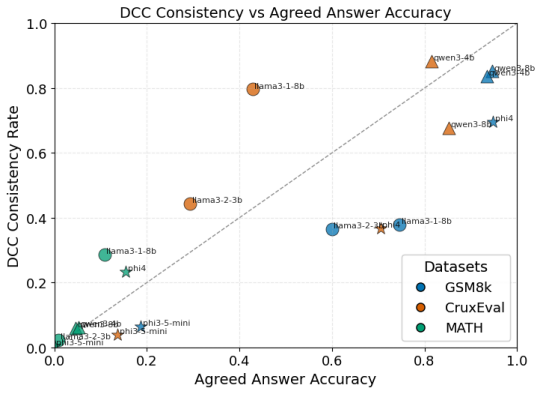


Figure 2: DCC metric versus accuracy at first sample.

We find that while DCC and accuracy exhibit some correlation, the trade-off between them differs considerably across tasks and models (Fig. 2). On GSM8K, most models achieve higher accuracy than DCC, suggesting that they are particularly prone to memorization shortcuts on this benchmark and that strong accuracy does not necessarily reflect true reasoning capability. In contrast, CruxEval sees the opposite trend, with generally better DCC than accuracy, suggesting that benchmark performance on this dataset is not yet saturated by memorization. Finally, on MATH, both metrics remain consistently low across models, reflecting its overall difficulty: producing two consistent answers is already challenging, and constructing coherent interventions and accurately predicting counterfactuals even more so.

The varying trade-offs between DCC and accuracy suggest that DCC captures a dimension of causal reasoning distinct from standard accuracy-based measures. DCC explicitly tests the model's ability to apply interventions and reason counterfactually, providing a measure of whether the model internally represents the causal structure of the problem. This makes DCC particularly valuable for identifying cases where a model appears accurate but is not reasoning causally, as well as for highlighting situations where a model is capable of causal reasoning even if it produces an initially incorrect answer. In this sense, DCC complements conventional accuracy metrics by emphasizing the process of reasoning rather than just the outcome.

5.2 DCC AS INFERENCE-TIME MODEL STEERING

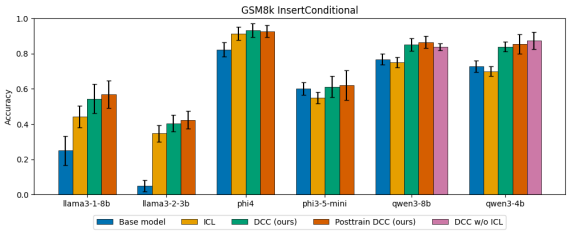
On counterfactual benchmarks, we compare the performance of the two DCC-based learning methods—the inference-time rejection sampling criterion (**DCC**) and the post-training reward (**post-train DCC**)—against the baselines described in Section 4.2. We find that in the RE-IMAGINE GSM8K setting, both DCC approaches substantially improve performance relative to the base model and ICL baselines for the `IrrelevantInfoHard` and `InsertConditional` mutations, with post-training DCC yielding slightly higher gains (Figures 3a, 3b).

Qwen-family models stand out as an exception on the `IrrelevantInfoHard` mutation, where in-context examples seem to strongly degrade performance (as shown by the lower performance of the ICL baseline compared to the base model). This negatively impacts DCC, which requires in-context examples to elicit the proper response format. Therefore, although DCC improves performance relative to the ICL baseline, the negative impact of the in-context examples outweighs gains from the consistency criterion itself. We hypothesize that this may be due to the established pattern of reasoning that Qwen uses in its signature thinking mode, which may be disrupted by in-context examples that follow a different logic pattern.

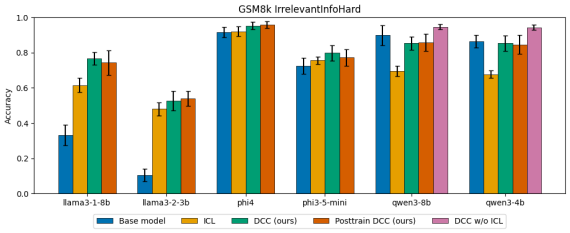
To explore this impact on Qwen, we also implement a version of inference-time DCC that provides the model with instructions for the three reasoning steps *without* any in-context examples (**DCC w/o ICL**). We find that this version significantly improves Qwen's performance on both RE-IMAGINE GSM8K datasets, outperforming not only standard DCC but also the base model and ICL baselines. This suggests that for models like Qwen, which are already adept at following their own reasoning patterns, it may be beneficial to test whether they can execute DCC without in-context examples, as this can lead to stronger results.

On RE-IMAGINE CruxEval (Fig. 3c), we find that DCC offers mixed results: Llama models benefit from DCC, while other model families see negative impacts. This degradation may stem from the higher complexity of the counterfactual CruxEval problems, which already contain a challenging counterfactual statement. Under these circumstances, models may struggle not only to execute and invert additional interventions but also to construct meaningful interventions to begin with (all while needing to produce accurate intermediate counterfactual predictions). Notably, in our analysis of DCC as a metric, we find that Llama models are also the only models to exhibit a much higher DCC relative to their accuracy on CruxEval (Fig. 2), suggesting that the DCC *metric* may help to identify models for which the DCC *criterion* or *reward* has the potential to improve performance.

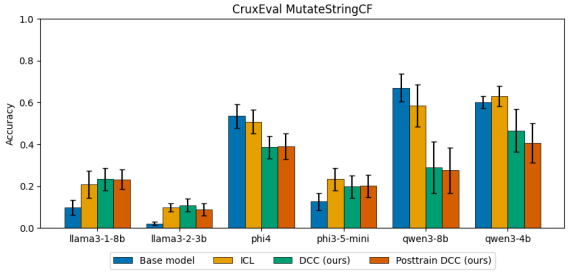
Finally, on standard GSM8K, DCC yields little to no improvement over ICL alone (Fig. 4a). This aligns with our expectation that extensive LLM memorization of GSM8K limits the benefits of causal reasoning, as guiding models away from shortcut-based solutions can obscure potential gains. In contrast, standard CruxEval—though not explicitly counterfactual—does show DCC-driven improvements for certain models (Llama-family and Phi-3.5 Mini), as seen in Fig. 4b. This pattern is consistent with our earlier observation in Section 5.1 that CruxEval exhibits higher DCC metrics than accuracy, suggesting that it is less saturated by memorization and thus leaving more room for DCC to guide models through causal reasoning steps.



(a) RE-IMAGINE GSM8K (`IC`)

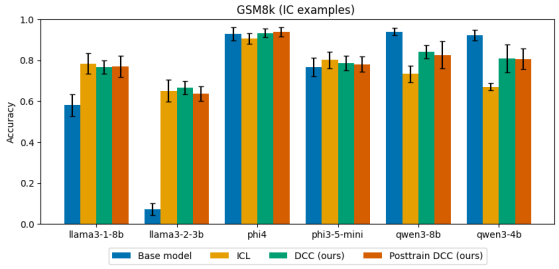


(b) RE-IMAGINE GSM8K (`IIH`)

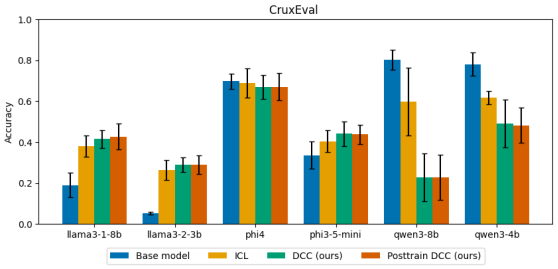


(c) RE-IMAGINE CruxEval (`MSCF`)

Figure 3: Performance comparisons on counterfactual reasoning benchmarks.



(a) Original GSM8K (with `InsertConditional`-style in-context examples for ICL and DCC methods).



(b) Original CruxEval (with `MutateStringCF`-style in-context examples for ICL and DCC methods).

Figure 4: Performance comparisons on standard reasoning benchmarks.

Overall, we find that DCC improves performance primarily in settings requiring explicit counterfactual reasoning. Its impact diminishes when models lack capacity or when benchmarks are saturated by memorization, suggesting that DCC's current utility as a causal reasoning learning method depends on both the reasoning demands of the task and the model's baseline competence.

5.3 Why and when does DCC work?

In this section, we conduct qualitative analysis on example DCC traces to better understand the success and failure cases of our approach. In Appendix C, we provide example traces from Phi-4, Qwen3 8B, and Llama 3.2 3B. Additional error analysis can be found in Appendix B.

In the Phi-4 traces, we see that when DCC holds (Lst. 2), the model not only answers the question correctly but also successfully introduces and inverts the intervention, completing both components required for causal reasoning: intervention and counterfactual prediction. When DCC does not hold (Lst. 3), the mismatch between the first and third answers is due to incorrect inversion of the intervention in the third question.

Turning to the Qwen traces, we see that when in-context examples are provided as part of the system prompt, DCC does not hold (Lst. 4), and the model—similarly to Phi-4—incorrectly inverts the intervention in the third question. Notably, the reasoning trace departs from Qwen's typical iterative "thinking" style, where the model revisits and double-checks its answers. When no in-context examples are included in the system prompt, DCC does hold (Lst. 5), and Qwen's typical reflective thinking process is restored. In fact, iterating on its response allows Qwen to catch an initial error when inverting the intervention in the third question. These traces not only demonstrate how DCC works but also help illustrate why Qwen-family models can be negatively impacted by ICL, as the signature iterative thinking process that supports their strong performance may be compromised when following the strict format of in-context examples.

These examples highlight several important points. First, DCC is indeed measuring the two core components of causal reasoning we identify in this paper: causal intervention and counterfactual prediction. DCC holds when both elements are executed correctly and does not hold when at least one element (e.g., intervention) is executed incorrectly. Second, both models exhibit instances of correct and incorrect causal reasoning and answers. As a metric, DCC can measure the proportion of samples for which the model reasons correctly, i.e., how consistent a model is in applying causal reasoning. As a rejection sampling method or reward, DCC can reject or discourage responses where the model exhibits failures in causal reasoning.

6 Related Work

6.1 Evaluating causal reasoning

Language models have long been established to struggle with counterfactual (*"what if?"*) reasoning. Early benchmarks such as WIQA and IfQA (Tandon et al., 2019; Yu et al., 2023) revealed difficulties in handling hypothetical perturbations and presuppositions. Recent efforts have further highlighted this challenge: CounterBench introduces a suite of formal counterfactual queries showing near-random performance

across LLMs (Chen et al., 2025). In concurrent work, RE-IMAGINE generates counterfactual variants across Pearl's causal hierarchy to disentangle genuine reasoning from memorization (Xu et al., 2025).

Alongside benchmarks, explicit metrics have emerged to assess reasoning quality. Chain-of-thought coherence has been evaluated via knowledge graph grounding (Nguyen et al., 2024), while formalized logical and deductive consistency metrics reveal brittleness in multi-hop reasoning (Liu et al., 2024; Pandey et al., 2025). These works mark a shift towards evaluating not only what answer an LLM gives but also how sound its reasoning process is. DCC fits into this landscape as a targeted metric of causal consistency.

6.2 COUNTERFACTUAL DATA GENERATION

Another line of research treats LLMs as structural causal models, enabling the generation of counterfactual outputs through controlled interventions (Ravfogel et al., 2024; Chatzi et al., 2024). These methods alter specific attributes of texts while keeping other factors fixed, facilitating the creation of contrastive natural language examples for analyzing causal effects and model biases. While DCC produces counterfactual text in its reasoning trace, its goal is not to generate counterfactual examples but rather to evaluate a model's internal consistency, with counterfactual scenarios simply a byproduct of this procedure.

6.3 INFERENCE-TIME VERIFICATION

A variety of strategies have been proposed to improve the correctness and consistency of LLM reasoning—often at inference time by sampling multiple outputs or enforcing internal agreement. Self-consistency based decoding (Wang et al., 2022; 2024) samples diverse chains-of-thought and then selects the final answer by majority vote. Manakul & Gales (2023) propose SelfCheckGPT, which flags a response as a potential hallucination if sampled responses diverge. More recently, Kim & Hwang (2025) introduce a targeted prompting technique for temporal reasoning that automatically generates temporally counterfactual variants of a query (e.g. flipping "before" to "after" in an event description) and enforces consistency constraints on the answers. Finally, consistency has also been used as a weak supervision signal in model training. Test-Time Reinforcement Learning (TTRL) rewards models for aligning with their own majority predictions, improving performance without labeled data (Zuo et al., 2025).

While we focus on two specific ways in which DCC can be used to improve performance in this paper—an inference-time rejection sampling criterion and a reinforcement learning reward—DCC is also complementary to these other inference-time strategies. For instance, DCC can be directly integrated into a self-consistency or search-based decoder to filter or re-rank answers based on their counterfactual consistency.

7 CONCLUSION

In this paper, we introduce double counterfactual consistency, a lightweight method for evaluating and guiding causal reasoning in large language models. Unlike benchmarks that rely on labeled counterfactual data, DCC directly tests whether a model can execute interventions and predict counterfactual outcomes, offering an unsupervised measure of causal reasoning ability. Across a range of reasoning benchmarks, we find that DCC captures aspects of causal robustness not reflected in standard accuracy metrics, revealing that models with comparable base accuracy can differ substantially in their double counterfactual consistency.

We further find that DCC can serve as a practical inference-time criterion, improving performance when used for rejection sampling or post-training rewards. At the same time, empirical findings underscore that to succeed as a learning method, DCC relies on a certain existing baseline capacity for intervention and counterfactual prediction in the model. We suggest that refining DCC as a post-training alignment signal may yield larger and more robust performance improvements in future work.

Taken together, our findings highlight that current benchmarks and evaluation practices can conflate surface-level reasoning with causal understanding. DCC offers a scalable, model-agnostic way to disentangle these abilities, paving the way for more fine-grained evaluation of causal reasoning in LLMs and for new methods that explicitly target this capacity at inference or training time.

REFERENCES

Ivi Chatzi, Nina Corvelo Benz, Eleni Straitouri, Stratis Tsirtsis, and Manuel Gomez-Rodriguez. Counterfactual token generation in large language models. *arXiv preprint arXiv:2409.17027*, 2024.

Yuefei Chen, Vivek K Singh, Jing Ma, and Ruxiang Tang. Counterbench: A benchmark for counterfactuals reasoning in large language models. *arXiv preprint arXiv:2502.11008*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Javier González and Aditya V. Nori. Does reasoning emerge? examining the probabilities of causation in large language models. 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. 117737–117761. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d5a1f97d2b922da92e880d13b7d2bf02-Paper-Conference.pdf.

Alex Gu, Baptiste Roziere, Hugh James Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida Wang. CRUXEval: A benchmark for code reasoning, understanding and execution. In Ruslan Salakhutdinov, Zico Kolte91 06 28 49 87 85r, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16568–16621. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/gu24c.html.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle Cai, Hui Yuan, Runzhe Wang, Yue Wu, Ming Yin, Shange Tang, Yangsibo Huang, Chi Jin, Xinyun Chen, Chiyuan Zhang, and Mengdi Wang. MATH-perturb: Benchmarking LLMs' math reasoning abilities against hard perturbations. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=OZy70UggXr.

Alihan Hüyük, Xinnuo Xu, Jacqueline Maasch, Aditya V. Nori, and Javier González. Reasoning elicitation in language models via counterfactual feedback, 2025. URL https://arxiv.org/abs/2410.03767.

Jongho Kim and Seung-won Hwang. Counterfactual-consistency prompting for relative temporal understanding in large language models. *arXiv preprint arXiv:2502.11425*, 2025.

Yinhong Liu, Zhijiang Guo, Tianya Liang, Ehsan Shareghi, Ivan Vulić, and Nigel Collier. Aligning with logic: Measuring, evaluating and improving logical consistency in large language models. *arXiv e-prints*, pp. arXiv–2410, 2024.

Potsawee Manakul and Mark Gales. Selfcheckgpt: Detecting hallucinations without external knowledge. In *Proceedings of ACL*, 2023.

Minh-Vuong Nguyen, Linhao Luo, Fatemeh Shiri, Dinh Phung, Yuan-Fang Li, Thuy-Trang Vu, and Gholamreza Haffari. Direct evaluation of chain-of-thought in multi-hop reasoning with knowledge graphs. *arXiv preprint arXiv:2402.11199*, 2024.

Atharva Pandey, Kshitij Dubey, Rahul Sharma, and Amit Sharma. Deduce: Deductive consistency as a framework to evaluate llm reasoning. *arXiv preprint arXiv:2504.07080*, 2025.

Shauli Ravfogel, Anej Svete, Vésteinn Snæbjarnarson, and Ryan Cotterell. Gumbel counterfactual generation from language models. *arXiv preprint arXiv:2411.07180*, 2024.

Donald B Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American statistical Association*, 100(469):322–331, 2005.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Niket Tandon, Bhavana Dalvi Mishra, Antoine Bosselut, Peter Clark, and Yejin Choi. Wiqa: A dataset for "what if" reasoning over procedural text. In *Proceedings of EMNLP*, 2019.

Ante Wang, Linfeng Song, Ye Tian, Baolin Peng, Lifeng Jin, Haitao Mi, Jinsong Su, and Dong Yu. Self-consistency boosts calibration for math reasoning. *arXiv preprint arXiv:2403.09849*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. arxiv. *Preprint posted online March*, 28, 2024.

Xinnuo Xu, Rachel Lawrence, Kshitij Dubey, Atharva Pandey, Risa Ueno, Fabian Falck, Aditya V. Nori, Rahul Sharma, Amit Sharma, and Javier Gonzalez. Re-imagine: Symbolic benchmark synthesis for reasoning evaluation. In *International Conference on Machine Learning*, 2025.

Mo Yu, Yichong Zhang, and Danqi Chen. Ifqa: Counterfactual reasoning for open-domain question answering. In *Proceedings of ACL*, 2023.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

Parameter	Value
Learning rate	5×10^{-6}
Optimizer	AdamW
Num epochs	4
Num generations per prompt	32
Batch size	64
Gradient accumulation steps	1
Temperature / top-p	0.6 / 0.95
Max completion length	1024
LoRA rank / alpha	16 / 32
Max grad norm	0.1
Precision	bf16

Table 1: Key training hyperparameters.

Dataset	License
GSM8K	MIT
CruxEval	MIT
MATH	MIT
RE-IMAGINE GSM8K	-
RE-IMAGINE CruxEval	-

Table 2: Dataset details.

A ADDITIONAL EXPERIMENT DETAILS

A.1 POST-TRAINING REGIME

When post-training with DCC reward, we fine-tune LoRA modules (rank 16, dropout 0.0, target modules: projection + MLP layers) using group-robust policy optimization (GRPO). Core hyperparameters are in Table 1. Models are loaded via Unsloth in 4-bit and optimized with 8-bit paged AdamW. Context length derives from system prompt length + (optional) exemplars + reserve (512) plus completion budget (up to 1024 new tokens).

A.2 EVALUATION

Results in Section 5.1 are computed over the full test split of the benchmark datasets. Results in Section 5.2 are computed over 15 batches of 64 samples drawn (with replacement) from the test split of the benchmark datasets. Dataset license details are given in Table 2.

Answer Parsing Completions use an XML scaffold:

```
<reasoning> ... </reasoning>
<answer> ... </answer>
```

For DCC-type prompts, three such segments are concatenated; the parser returns either a single normalized answer or a triplet. Equality is strict post-normalization (string-level); numerically equivalent but differently formatted expressions may be mismatched.

A.3 COMPUTING RESOURCES

All experiments were conducted on NVIDIA A100 40 GB and 80 GB GPUs. We estimate the number of GPU hours used to be around 1500.

B ERROR ANALYSIS

We elaborate on several potential hypotheses for why DCC can fail. To provide evidence for these explanations, we conduct detailed error analysis on a Llama 3.2-3b reasoning trace (Listing 6) after 5 epochs of DCC reward post-training on RE-IMAGINE GSM8K (`InsertConditional` mutation).

Our first hypothesis is that in challenging settings, models may lack even the base capacity to consistently answer questions correctly. This hypothesis is supported by this reasoning trace, where we have a smaller capacity model faced with a relatively challenging initial question. We see that the model reasons incorrectly about even this initial question, concluding that $55 \geq 63$, and therefore answering incorrectly with 55 rather than 0.

Our second hypothesis is that models can struggle to construct meaningful interventions under challenging conditions, which also manifests in this trace. While the model successfully introduces an intervention for the second question and answers it correctly, it incorrectly inverts the intervention when constructing the third question—in fact merely restating the intervention without inverting it at all.

Our third hypothesis is that without proper safeguards like early stopping, degradation from reward shortcutting can occur as the model learns that the first answer should always equal the last. While we would typically stop training earlier, we allowed training to go to 5 epochs here for more informative error analysis. Under these atypical training conditions, shortcutting does appear in this example, supporting our hypothesis. In the third question, the model reasons correctly that it should reveal a random number rather than Randolph's true age. However, the model disregards this reasoning and then incorrectly repeats the answer from the first question.

C EXAMPLE DCC TRACES

Listing 2 (DCC holds) Phi-4 DCC trace on GSM8K

```
1. Question: Samwell owns an appliances store. For this week, one-fourth of
↪  their sales are smart TVs, one-eighth are analog TVs, and the rest are
↪  OLED TVs. If they were able to sell a total of 40 TVs, how many OLED TVs
↪  were they able to sell?

<reasoning>
1. Total TVs sold = 40
2. Smart TVs sold = 1/4 of 40 = 10
3. Analog TVs sold = 1/8 of 40 = 5
4. OLED TVs sold = Total TVs - (Smart TVs + Analog TVs) = 40 - (10 + 5) = 25
</reasoning>
<answer>
25
</answer>

2. Now consider a counterfactual version of this question, where the following
↪  intervention is made: Suppose that Samwell's store sells an additional 5
↪  smart TVs due to a special promotion, while the total sales remain at 40
↪  TVs. Calculate how many OLED TVs they were able to sell under this new
↪  condition.
<reasoning>
1. Total TVs sold = 40
2. Smart TVs sold = 1/4 of 40 + 5 (promotion) = 10 + 5 = 15
3. Analog TVs sold = 1/8 of 40 = 5
4. OLED TVs sold = Total TVs - (Smart TVs + Analog TVs) = 40 - (15 + 5) = 20
</reasoning>
<answer>
20
</answer>

3. Now consider a counterfactual version of the *counterfactual* question,
↪  where the following change intervenes on the previous intervention to
↪  restore it to its original state: Suppose that the promotion is canceled,
↪  so no additional smart TVs are sold, and the original sales distribution
↪  is restored while maintaining the total sales at 40 TVs. Calculate how
↪  many OLED TVs they were able to sell.
<reasoning>
1. Total TVs sold = 40
2. Smart TVs sold = 1/4 of 40 = 10 (promotion canceled)
3. Analog TVs sold = 1/8 of 40 = 5
4. OLED TVs sold = Total TVs - (Smart TVs + Analog TVs) = 40 - (10 + 5) = 25
</reasoning>
<answer>
25
</answer>
```

Listing 3 (DCC does not hold) Phi-4 DCC trace on GSM8K

```
1. Question: When Sophie watches her nephew, she gets out a variety of toys
↪  for him. The bag of building blocks has 31 blocks in it. The bin of
↪  stuffed animals has 8 stuffed animals inside. The tower of stacking rings
↪  has 9 multicolored rings on it. Sophie recently bought a tube of bouncy
↪  balls, bringing her total number of toys for her nephew up to 62. How many
↪  bouncy balls came in the tube?

<reasoning>
1. Total toys from building blocks = 31
2. Total toys from stuffed animals = 8
3. Total toys from stacking rings = 9
4. Total toys before adding bouncy balls = 31 + 8 + 9 = 48
5. Total toys after adding bouncy balls = 62
6. Number of bouncy balls = 62 - 48 = 14
</reasoning>
<answer>
14
</answer>

2. Now consider a counterfactual version of this question, where the following
↪  intervention is made: Suppose that the total number of toys Sophie has is
↪  70, instead of 62; calculate how many bouncy balls came in the tube.
<reasoning>
1. Total toys from building blocks = 31
2. Total toys from stuffed animals = 8
3. Total toys from stacking rings = 9
4. Total toys before adding bouncy balls = 31 + 8 + 9 = 48
5. Total toys after adding bouncy balls = 70
6. Number of bouncy balls = 70 - 48 = 22
</reasoning>
<answer>
22
</answer>

3. Now consider a counterfactual version of the *counterfactual* question,
↪  where the following change intervenes on the previous intervention to
↪  restore it to its original state: Suppose that the total number of toys
↪  Sophie has remains 62, but she initially counted 2 extra stuffed animals
↪  by mistake, making the initial count of stuffed animals 6 instead of 8;
↪  calculate how many bouncy balls came in the tube.
<reasoning>
1. Total toys from building blocks = 31
2. Corrected total toys from stuffed animals = 6 (initially counted as 8)
3. Total toys from stacking rings = 9
4. Total toys before adding bouncy balls = 31 + 6 + 9 = 46
5. Total toys after adding bouncy balls = 62
6. Number of bouncy balls = 62 - 46 = 16
</reasoning>
<answer>
16
</answer>
```

Listing 4 (DCC does not hold) Qwen3-8b DCC (with ICL) trace on GSM8K

```
<<Pamela reapplies an ounce of sunscreen every hour she's outside.  Her
↪  sunscreen comes in 8-ounce bottles.  While on vacation, if she will be
↪  outside 4 hours a day over 8 days, how many bottles of sunscreen will she
↪  need to pack?>>

<reasoning>
1. Total hours outside = 4 hours/day × 8 days = 32 hours.
2. Total ounces of sunscreen needed = 32 ounces (since she uses 1 ounce per
↪  hour).
3. Each bottle contains 8 ounces, so the number of bottles needed = 32 ounces
↪  ÷ 8 ounces/bottle = 4 bottles.
</reasoning>
<answer>
4
</answer>

2. Now consider a counterfactual version of this question, where the following
↪  intervention is made: Suppose that Pamela decides to reapply sunscreen
↪  every two hours instead of every hour; she will still be outside 4 hours a
↪  day over 8 days. Calculate how many bottles of sunscreen she will need to
↪  pack.
<reasoning>
1. Total hours outside = 4 hours/day × 8 days = 32 hours.
2. Total reapplications = 32 hours ÷ 2 hours/application = 16 applications.
3. Total ounces of sunscreen needed = 16 ounces (since she uses 1 ounce per
↪  application).
4. Each bottle contains 8 ounces, so the number of bottles needed = 16 ounces
↪  ÷ 8 ounces/bottle = 2 bottles.
</reasoning>
<answer>
2
</answer>

3. Now consider a counterfactual version of the *counterfactual* question,
↪  where the following change intervenes on the previous intervention to
↪  restore it to its original state: Suppose that Pamela decides to reapply
↪  sunscreen every hour, but each bottle now contains 16 ounces instead of 8
↪  ounces. Calculate how many bottles of sunscreen she will need to pack.
<reasoning>
1. Total hours outside = 4 hours/day × 8 days = 32 hours.
2. Total ounces of sunscreen needed = 32 ounces (since she uses 1 ounce per
↪  hour).
3. Each bottle now contains 16 ounces, so the number of bottles needed = 32
↪  ounces ÷ 16 ounces/bottle = 2 bottles.
</reasoning>
<answer>
2
</answer>
```

Listing 5 (DCC holds) Qwen3-8b DCC (without ICL) trace on GSM8K

```
<<Pamela reapplies an ounce of sunscreen every hour she's outside.  Her sunscreen comes in 8-ounce
↪  bottles.  While on vacation, if she will be outside 4 hours a day over 8 days, how many bottles
↪  of sunscreen will she need to pack?>>

<think>
Completion: Okay, let's see. The problem is about Pamela needing sunscreen. She reapplies an ounce
↪  every hour she's outside. The sunscreen comes in 8-ounce bottles. She's going to be outside 4
↪  hours a day for 8 days. How many bottles does she need?

First, I need to calculate the total amount of sunscreen she'll use. She uses 1 ounce per hour. So,
↪  4 hours a day times 8 days gives 32 hours total. That means she needs 32 ounces. Since each
↪  bottle is 8 ounces, divide 32 by 8. That's 4 bottles. So the answer is 4.

Now, the counterfactual question. Let's say she reapplies every 2 hours instead of every hour. Then,
↪  she uses 0.5 ounces per hour. Wait, no. If she reapplies every 2 hours, then in 4 hours, she
↪  would apply 2 times. So total ounces would be 2 times per day. Over 8 days, that's 16
↪  applications. Each application is 1 ounce, so total is 16 ounces. Then, 16 divided by 8 is 2
↪  bottles. So the answer here is 2.

Then the next step is to reverse that intervention. If we go back to the original scenario where she
↪  reapplies every hour, but somehow the 4 days are changed back to 8 days. Wait, the original
↪  question had 8 days. Wait, the counterfactual was changing the reapplied time. So to reverse
↪  that, maybe we need to adjust the hours. Wait, the original question had 4 hours a day over 8
↪  days. If the first counterfactual was changing the reapplied time to every 2 hours, then the
↪  second counterfactual would need to change that back. Wait, maybe the second counterfactual is
↪  changing the reapplied time back to every hour, but adjusting something else. Wait, the user's
↪  instruction says the second counterfactual is an intervention on the previous one. So the first
↪  counterfactual was changing the reapplied time to every 2 hours. Then, the second counterfactual
↪  would be changing that intervention back. So if the first counterfactual was "she reapplies
↪  every 2 hours", then the second would be "she reapplies every hour again". But then the answer
↪  would be the original 4. But the user wants the second counterfactual to restore the original
↪  question. So maybe the first counterfactual was changing the number of days or hours. Wait,
↪  maybe I need to re-express.

...

Alternatively, maybe the first counterfactual is changing the number of days. For example, if she's
↪  outside for 4 days instead of 8. Then total hours is 4*4=16, 16/8=2 bottles. Then the second
↪  counterfactual would change the days back to 8, so 8*4=32, 32/8=4. So the answer would be 4
↪  again. So each step's answer is an integer. So the steps would be:

Original: 4 bottles.

First counterfactual: 2 bottles.

Second counterfactual: 4 bottles.

But I need to make sure that each answer is an integer. So that works. So the first answer is 4, the
↪  second is 2, the third is 4. So the steps are correct.
</think>

<reasoning>
1. The question asks for the number of 8-ounce sunscreen bottles Pamela needs if she uses 1 ounce
↪  per hour, is outside 4 hours a day over 8 days. Total hours: 4 * 8 = 32. Total ounces: 32.
↪  Bottles: 32 / 8 = 4. Answer: 4.

2. Counterfactual: Suppose she reapplies every 2 hours instead of every hour. Now, she uses 0.5
↪  ounces per hour. Total hours: 4 * 8 = 32. Total ounces: 32 * 0.5 = 16. Bottles: 16 / 8 = 2.
↪  Answer: 2.

3. Counterfactual of the counterfactual: Restore the original reapplication rate (1 ounce per hour)
↪  but reduce the number of days to 4. Total hours: 4 * 4 = 16. Total ounces: 16. Bottles: 16 / 8 =
↪  2. Wait, this doesn't restore the original state. To restore the original, adjust the hours per
↪  day back to 4 and days to 8. But the intervention here is changing the reapplication rate. To
↪  reverse, set reapplication rate back to 1 ounce/hour and adjust days to 8. Total hours: 4 * 8 =
↪  32. Bottles: 32 / 8 = 4. Answer: 4.
</reasoning>

<answer>
4
</answer>

<answer>
2
</answer>

<answer>
4
</answer>
```

Listing 6 (Error analysis) Llama 3.2-3b post-train DCC trace on RE-IMAGINE GSM8K with `InsertConditional` mutation

```
1. Randolph is 5 years older than Sydney. Sydney is twice as old as Sherry.
↪  Sherry is 25. Suppose that Randolph will only reveal his age if he is at
↪  least 63 years older than Sydney; if this condition is met, calculate his
↪  age as usual, otherwise assume his age is 0. How old is randolph?

<reasoning>
Sherry is 25 years old. Sydney is twice as old as Sherry, so Sydney is 2 × 25
↪  = 50 years old. Randolph is 5 years older than Sydney, so Randolph is 50 +
↪  5 = 55 years old. Since 55 >= 63, Randolph meets the condition and will
↪  reveal his age. Therefore, his age is 55.
</reasoning>
<answer>
55
</answer>

2. Now consider a counterfactual version of this question, where the following
↪  intervention is made: Suppose that Randolph will reveal his age if he is
↪  at least 63 years older than Sydney; if this condition is not met, he will
↪  reveal a random number between 0 and 63..
How old is randolph?

<reasoning>
Since Sherry is 25, Sydney is 50, and Randolph is 55, the condition is met.
↪  However, the new rule requires us to reveal a random number, which is
↪  irrelevant to determining Randolph's age. Therefore, we cannot determine
↪  Randolph's age.
</reasoning>
<answer>
cannot be determined
</answer>

3. Now consider a counterfactual version of the *counterfactual* question,
↪  where the following change intervenes on the previous intervention to
↪  restore it to its original state: Suppose that Randolph will reveal a
↪  random number between 0 and 63 if the condition of being at least 63 years
↪  older than Sydney is not met; otherwise, he will reveal his age as usual..
How old is randolph?

<reasoning>
Since Sherry is 25, Sydney is 50, and Randolph is 55, the condition is met.
↪  However, the new rule requires us to reveal a random number, which is
↪  irrelevant to determining Randolph's age. Since the condition is met,
↪  Randolph's age is 55.
</reasoning>
<answer>
55
</answer>
```