A Formal Verification Framework for LLM-Generated Causal Expressions

Anonymous ACL submission

Abstract

Causal reasoning is a critical aspect in human judgment, yet large language models (LLMs) often fail to distinguish between correlation and causation when answering natural language questions. While existing benchmarks primarily assess factual correctness in causal QA tasks, they do not evaluate whether model outputs are causally coherent or formally valid. In this work, we propose DoVerifier, a symbolic evaluation framework that assesses whether LLM-generated answers can be correctly formalized as causal expressions using do-calculus semantics. Our approach translates LLM outputs into structured forms such as $P(Y \mid \mathrm{do}(X), Z)$, compares them against known ground-truth assumptions or causal graphs, and identifies common reasoning failures such as misinterpreting interventions as observations. We further demonstrate that this formalization layer enables symbolic feedback, which can guide LLMs to revise incorrect outputs and improve overall answer quality.[1]

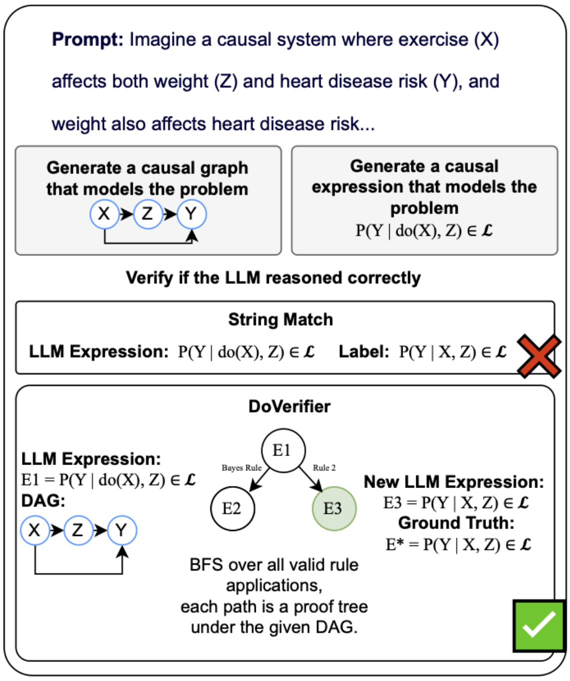

Figure 1: Our symbolic verifier checks whether a model-generated causal expression is semantically equivalent to the ground truth under a given DAG. Unlike string match, it explores all valid derivations using do-calculus and probability rules to identify formal equivalence.

1 Introduction

Causal reasoning lies at the core of human intelligence. Unlike mere pattern recognition, it enables us to reason about interventions, explain effects, and predict outcomes under hypothetical scenarios. As large language models (LLMs) (OpenAI, 2024; Team, 2025; DeepSeek-AI, 2025) are increasingly deployed in scientific, medical, and policy-related domains, the ability to generate and interpret causal claims is no longer optional—it is critical (Doshi-Velez and Kim, 2017). An LLM that can distinguish between correlation and causation could support tasks ranging from experimental design to scientific hypothesis generation.

Recent causal reasoning benchmarks such as CLadder (Jin et al., 2023) and CausalBench (Wang, 2024) have begun to evaluate LLMs on causal question answering. However, these efforts primarily focus on surface-level correctness: whether the model's answer matches a gold string or produces the right outcome in simple scenarios. While useful, these metrics fail to capture a more fundamental question: *does the model's output represent a valid causal expression under formal semantics?*

This gap is especially significant because the verification of causal statements is not as simple as evaluating mathematical statements. In mathematical formalization tasks, models can often be evaluated by plugging in values or checking numerical correctness (Gao et al., 2025; Fan et al., 2024; Cobbe et al., 2021; Hendrycks et al., 2021).

[1]The full code of DoVerifier, along with the evaluation code and synthetic dataset, will be released.

However, in causal inference, as shown in Figure 1, we rarely know the full joint distribution $P(\cdot)$; the ground truth is defined not by observed values, but by derivability under a causal graph using the rules of do-calculus (PEARL, 1995). This makes causal verification fundamentally symbolic: an expression like $P(Y \mid \mathrm{do}(X))$ must be judged valid based on its formal relation to a DAG and other expressions—not via simulation or numeric output.

To our knowledge, no existing work has proposed a general-purpose symbolic verifier for LLM-generated causal expressions. Even the most advanced causal benchmarks (Wang, 2024; Ma, 2025) still focus on the final answer, and evaluating the reasoning steps overlooks semantic equivalences despite syntactical differences. Existing metrics conflate syntactic similarity with semantic equivalence and offer no way to detect subtle logical errors, such as confusing observations with interventions. For example, as the case shown in Figure 1, the LLM-generated expression can be simplified using the rules of do-calculus from an observational form to an interventional form. This limits the precision of causal QA evaluation and hinders progress toward more reliable reasoning agents.

To address this concern, in this paper, we introduce a formal verification framework, DoVerifier, that evaluates the causal validity of LLM outputs. As illustrated in Figure 1, our method translates model-generated answers into structured symbolic expressions and verifies whether these expressions are derivable from known assumptions using the do-calculus and probability rules. This enables a rigorous test of whether a response is causally coherent, regardless of its surface form. By introducing symbolic verification into LLM evaluation, our work opens the door to more trustworthy large language models in the causality domain.

The main contributions of this paper are 3-fold:

- We define causal equivalence as a reachability problem in a derivation graph, and develop a sound and complete proof search algorithm based on breadth-first traversal with our verifier DoVerifier.
- We show that DoVerifier outperforms standard string-based metrics in both synthetic and real datasets, recovering semantically correct answers that would otherwise be marked incorrect.
- We demonstrate a symbolic feedback loop that uses the verifier's outputs to guide model corrections, improving causal accuracy without access to ground-truth answers.

2 Related Work

Causal QA and LLMs Recent benchmarks evaluate large language models (LLMs) on their ability to answer causal questions expressed in natural language. CLadder (Jin et al., 2023) and CausalBench (Wang, 2024) present standardized datasets of associational, interventional, and counterfactual queries grounded in causal graphs. Evaluation typically hinges on whether a model's output matches a gold-standard string, sometimes allowing paraphrasing or token-level similarity. While these approaches assess *factual correctness*, they lack a formal guarantee of *causal validity* (Jin et al., 2023; Bondarenko et al., 2022; Joshi et al., 2024). For instance, a model that outputs $P(Y \mid X)$ instead of $P(Y \mid \mathrm{do}(X))$ may still be marked correct—despite the two being causally distinct. These benchmarks do not assess whether an expression is *derivable* from the assumed graph using valid causal rules.

This paper addresses this gap by introducing a symbolic verification framework that checks whether a model output is derivable from known assumptions via the do-calculus and probability theory. This enables principled evaluation of causal soundness beyond surface form. For example, when asked to formalize "Does X cause Y?" in a graph where X affects Y through a mediator Z, models often produce $P(Y \mid X) > P(Y)$—an associational claim—rather than the correct interventional query $P(Y \mid \mathrm{do}(X)) \neq P(Y)$ (Chen et al., 2024). Such conceptual errors are undetected by benchmarks that assess only final answers.

Formal Verification and Symbolic Inference The causal inference community has long relied on do-calculus (PEARL, 1995) and probability theory to determine whether a causal query is identifiable from observational data. Classical identifiability algorithms (Shpitser and Pearl, 2008) and modern tools like `dosearch` (Tikka et al., 2021) formalize this process as a search over valid derivations. However, these tools are designed to compute a causal effect from known inputs—not to verify whether two arbitrary expressions are equivalent or whether a generated formula is consistent with a causal graph. However, this is significant to evaluate the multi-step reasoning process in causal reasoning. Another line of work, like Sheth et al. (2025), checks if answers align with predefined causal graphs but relies on template matching rather than formal derivations and cannot handle expressions involving do-calculus transformations. To push the frontier of such verification, in this paper, DoVerifier proposes to reframe it as a symbolic proof problem: given a candidate expression, it checks whether the expression is logically derivable from assumptions via valid transformations, enabling both evaluation and error diagnosis in model-generated outputs.

Formalization in Mathematical Reasoning Efforts in mathematical reasoning have primarily focused on verifying answers to quantitative problems. For instance, Hendrycks et al. (2021) evaluates LLMs on math competition problems, while Glazer et al. (2024) investigates symbolic solvers for arithmetic tasks. To further validate intermediate reasoning steps, another line of work (Ren et al., 2025; Wang et al., 2024) resorts to formal math descriptions (de Moura and Ullrich, 2021; Nipkow et al., 2002) that facilitate the step-wise consistency inspection. Although it is promising to *formalize* a math problem (AlphaProof and teams, 2024; Lin et al., 2025), checking its semantic correctness is found crucial yet under evolving (Lu et al., 2024; Xin et al., 2025). Recent work in geometry (Murphy et al., 2024) and logic (Li et al., 2024) uses SMT solvers to assess logical equivalence between informal text and formal theorems. We draw inspiration from this paradigm but extend it to causal inference—where correctness is defined not by logical validity alone, but by derivability under the rules of do-calculus and a causal DAG.

Causal Evaluation Metrics Most existing evaluation metrics for causal QA—such as exact match, BLEU (Papineni et al., 2002), or token-level F1—focus on surface similarity between model outputs and reference answers (Jin et al., 2023; Bondarenko et al., 2022; Hu and Zhou, 2024). These metrics fail to capture semantic equivalence between expressions that differ syntactically but are derivationally identical under the do-calculus (Hu and Zhou, 2024). Our method addresses this limitation by treating equivalence as a formal proof problem. By comparing expressions based on symbolic derivability, we provide a more faithful and theoretically grounded measure of causal correctness.

This verification-based approach not only strengthens evaluation but also enables more transparent and trustworthy causal reasoning systems, which is critical for high-stakes domains like healthcare, policy, and science.

3 DoVerifier: Causal Symbolic Verification Framework

3.1 Preliminaries

As language models increasingly engage with causal reasoning tasks, proper evaluation requires verifying adherence to formal causal inference rules. Do-calculus provides the fundamental axioms for manipulating interventional distributions in causal graphs.

The Rules of do-calculus Our method will make use of the fundamental rules of the do-calculus (PEARL, 1995). Let X, Y, Z, and W be arbitrary disjoint sets of nodes in a causal directed acyclic graph (DAG) G. Following the notation of Pearl (2012), we denote $G_{\overline{X}}$ the graph obtained from G by removing all the edges pointing to the nodes in X and we denote $G_{\underline{X}}$ the graph obtained by deleting all the edges emerging from the nodes in X. Furthermore, we use $G_{\overline{X}\underline{Z}}$to represent the deletion of all edges that have X as a source or target.

Rule 1 (Insertion/deletion of observations):

$$P(y \mid \text{do}(x), z, w) = P(y \mid \text{do}(x), w) \quad \text{if } (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{X}}} \tag{1}$$

Rule 2 (Action/observation exchange):

$$P(y \mid \text{do}(x), \text{do}(z), w) = P(y \mid \text{do}(x), z, w) \quad \text{if } (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{X}\underline{Z}}} \tag{2}$$

Rule 3 (Insertion/deletion of actions):

$$P(y \mid \text{do}(x), \text{do}(z), w) = P(y \mid \text{do}(x), w) \quad \text{if } (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{XZ(W)}}} \tag{3}$$

where $Z(W)$ is the set of Z-nodes that are not ancestors of any W-node in $G_{\overline{X}}$. The notation $(Y \perp\!\!\!\perp Z \mid X, W)_G$ represents d-separation in graph G, meaning all paths between Y and Z are blocked by conditioning on X and W.

Probability Rules In addition to do-calculus rules, our method incorporates standard probability transformations, including Bayes' rule, the chain rule, and the law of total probability. These rules, combined with do-calculus, provide a complete system for verifying equivalence between causal expressions. We will denote the set of probability calculus rules as $\mathcal{P}$.

3.2 Definitions

Definition (Causal Expression Language) Let $\mathcal{L}_{\text{causal}}$ be the set of expressions of the form $P(Y \mid \mathbf{Z})$, where Y and elements of $\mathbf{Z}$ are either observed variables or do-interventions (i.e $\text{do}(X)$). Expressions are defined over a causal DAG G with a finite node set V.

Definition (Derivability) Given a DAG G, we say the causal expression $E_{\text{init}} \vdash_G E_{\text{target}}$ if the target expression E_{target} can be obtained from the initial expression E_{init} via a finite sequence of applications of the rules in do-calculus and probability theory, respecting the conditional independencies implied by G.

3.3 Method

We define a symbolic verification framework, DoVerifier, for assessing equivalence between causal expressions derived from natural language. We first provide a list of desired properties a good evaluator should have, listed in Appendix A. Given a causal DAG G (which may be generated by the model), and two expressions $E_{\text{init}}, E_{\text{target}} \in \mathcal{L}_{\text{causal}}$, DoVerifier determines whether E_{target} is derivable from E_{init} under the axioms of do-calculus and standard probability theory. The system operates by enumerating proof sequences through a structured search procedure. Implementation details are provided in Appendix B. The framework consists of the following components:

1. **Expression Parser.** Parses natural language or symbolic input into normalized expressions in $\mathcal{L}_{\text{causal}}$, including both observation $P(Y \mid X)$ and interventional $P(Y \mid \text{do}(X))$ forms.

2. **Transformation Engine.** Encodes inference rules drawn from do-calculus and probability calculus, which serve as rewrite rules over symbolic expressions.

3. **Proof Search Module.** Executes a breadth-first search over the space of derivable expressions, applying transformation rules to discover a finite derivation sequence from E_{init} to E_{target}, if one exists.

We model causal equivalence as a reachability problem in a derivation graph:

Proposition 3.1 (Derivation Graph)**.** *Let $E_{init} \in \mathcal{L}_{causal}$. Define a directed graph $S(E_{init})$ where:*

- *Each node is a unique causal expression derivable from E_{init};*
- *An edge $E \to E'$ exists if E' can be obtained from E by applying a single valid transformation.*

Then $S(E_{init})$ is a well-defined, finite-branching graph.

The branching factor is finite since the number of variables in G is finite and each transformation rule applies to bounded subsets (proof in Appendix C)[2]. Hence, the core decision problem is:

$$\texttt{Equiv}(E_{\text{init}}, E_{\text{target}}; G) \triangleq \exists \Pi E_{\text{init}} \xrightarrow{\Pi} E_{\text{target}} \tag{4}$$

where Π is admissible under G, reducing evaluation of semantic correctness to proof search under graph constraints.

Verification Algorithm Given a causal graph G, source expression E_{init}, target expression E_{target}, and maximum depth d, we present Algorithm 1 as an algorithm to verify if E_{init} and E_{target} are equivalent bounded by depth d.

This approach guarantees finding the shortest sequence of transformations if one exists within the depth limit, as stated in our main theorem that concerns the soundness and completeness of the verification algorithm:

Proposition 3.2 (Soundness & Completeness of Proof Search)**.** *Let G be a causal DAG, and let $E_{init}, E_{target} \in \mathcal{L}_{causal}$. If $E_{init} \vdash_G E_{target}$, then Algorithm 1 returns a valid proof sequence within depth d, for some finite d. Conversely, if no such derivation exists within depth d, Algorithm 1 returns* None.

Proof. Proved in Appendix C. ■

In addition, if no derivation exists between E_{init} and E_{target} with $k \leq d$ steps, breadth-first search

[2] Although the depth of the derivation graph could be infinite, we bound the depth of the derivation graph by some depth d.

Algorithm 1 Causal Expression Equivalence Verification

Initialize queue $Q \leftarrow [(E_{\text{init}}, [])]$ ▷ (expression, proof path π)
Initialize visited set $V \leftarrow \{E_{\text{init}}\}$
while Q not empty **do**
  $(E, \pi) \leftarrow Q$.dequeue()
  if $E = E_{\text{target}}$ **then**
    return π ▷ Found equivalence
  end if
  if $\mid \pi \mid < d$ **then**
    for each applicable rule r **do**
      $E' \leftarrow$ apply(r, E)
      if $E' \notin V$ **then**
        V.add(E')
        Q.enqueue$((E', \pi + [r]))$
      end if
    end for
  end if
end while
return None ▷ No equivalence found within depth d

(BFS) will terminate after exploring all expressions within depth d. Further practical considerations are explained in Appendix D

Separately, we can view DoVerifier as a logical system defined over a formal language $\mathcal{L}_{\text{causal}}$ equipped with a set of derivation rules $\mathcal{R}$ and a background model that is either provided or generated by an LLM G (the DAG). In this view, we distinguish between:

Syntactic entailment ($H \vdash_G A$)**:** A is derivable from hypothesis H using the symbolic transformation rules admissible under G bounded by depth d.

Semantic entailment ($H \models_G A$)**:** A is true in all causal models consistent with G in which all $H_i \in H$ are true.

We require our inference system to satisfy:

Soundness: If $H \vdash_G A$, then $H \models_G A$

Completeness: If $H \models_G A$, then $H \vdash_G A$

These are properties of the underlying logical system, which are satisfied due to the completeness of do-calculus for causal identifiability (PEARL, 1995) and the standard probability axioms. Thus, the semantic correctness of model outputs can be equivalently verified through syntactic derivation, which forms the basis of our verifier.

4 Experiments and Results

4.1 Synthetic Data Test

To verify the internal consistency of the verifier, we construct a synthetic dataset of over 10,000 expression pairs $(E_{\text{init}}, E_{\text{target}})$ such that E_{target} is provably derivable from E_{init} under a known DAG G. Each pair involves between 1–4 rule applications and includes randomized use of do-calculus and probability rules. A description of data samples is shown in Appendix E.

Sampling Procedure Let $V = \{v_1, \ldots, v_n\}$ be a finite set of variables, and let $G = (V, E)$ be a randomly sampled acyclic graph. We sample the directed edges independently as $\mathbb{P}(v_i \to v_j) = p$ for $i < j$ where $p \in (0, 1)$ is the edge probability, and the ordering ensures the graph is acyclic. In our experiments, we fix $n \leq 10$ and $p = 0.5$ to balance expressivity and tractability. We first construct

$$e_1 = P(Y \mid \text{do}(X_1), \ldots, \text{do}(X_k), Z_1, \ldots, Z_m) \tag{5}$$

where $Y \in V$ is chosen uniformly at random, a subset of $V \backslash \{Y\}$ is chosen as intervention variables $\{X_i\}$ and additional variables $\{Z_j\}$ are included as conditioning set as observation. To ensure structural diversity, the number of intervention variables X_i and observational variables Y_i is randomly chosen per sample, subject to DAG constraints. Then, we define a symbolic derivation process π consisting of a sequence of rule applications:

$$e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} \ldots \xrightarrow{r_n} e_{n+1} \tag{6}$$

where each $r_i \in \{\text{Rule 1, Rule 2, Rule 3}\} \cup \mathcal{P}$. Rule applications are randomized but constrained to only apply when valid under the conditional independencies induced by G. Then, we set $E_{\text{init}} = e_1$ and $E_{\text{target}} = e_{n+1}$.

Synthetic Data Performance Our symbolic verifier achieves 100% precision and recall under depth limit $d = 5$, demonstrating correctness of the derivation engine, while other methods such as string match, or token-level F1 performed poorly due to E_{init} and E_{target} being too distinct. Consider the following example:

$$\text{LLM output: } P(C \mid \text{do}(A), B) \tag{7}$$

$$\text{Ground Truth: } P(C \mid B) \tag{8}$$

Metric	Llama3.1-8b			Mistral-7B			Llama-3.1-8B-Instruct			Gemma-7b-it		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
String Match	1.0	0.57	0.72	1.0	0.58	0.73	1.0	0.88	0.93	1.0	0.80	0.89
DoVerifier (Ours)	1.0	**0.73**	**0.85**	1.0	**0.94**	**0.97**	1.0	**0.90**	**0.94**	1.0	**0.84**	**0.91**

Table 1: DoVerifier identifies more correct causal expressions than string match. Precision (P), Recall (R), and F1 scores for four LLMs evaluated on CLadder. Our method improves recall substantially while maintaining perfect precision, revealing semantically correct answers missed by string-based metrics.

With the causal DAG structure:

$$A \to B \quad B \to D \quad C \to D$$

DoVerifier proves equivalence through these steps:

1. Applied Rule 2: Convert $\text{do}(A)$ to observation $P(C \mid \text{do}(A), B) = P(C \mid A, B)$
2. Applied Rule 1: Remove A due to d-separation $P(C \mid A, B) = P(C \mid B)$

Thus, $P(C \mid \text{do}(A), B) = P(C \mid B)$ is verified as equivalent, which string matching and token-level F1 would have failed to recognize.

The experiment results show a key strength of our framework that it can correctly recognize when two expressions are equivalent under the rules of do-calculus and probability, even if they differ in formatting, variable order, or surface form.

4.2 LLM Causal Reasoning Test

Uncovering Missed Correct Answers We evaluate the ability of our symbolic verifier to improve the accuracy of large language model (LLM) evaluation in causal reasoning. Specifically, we ask: *Can our method recover correct answers that are missed by naive evaluation metrics?*

Evaluated Dataset and Models To investigate this, we use the **CLadder** benchmark (Jin et al., 2023), a suite of causal questions grounded in known DAGs. Each question is paired with a ground-truth answer expressed as a formal causal expression. We prompt `Llama-3-8B` (Grattafiori et al., 2024), `Llama-3-8BInstruct` (Grattafiori et al., 2024), `Mistral-7B` (Jiang et al., 2023), and `Gemma-7B-IT` (Team et al., 2024) to answer these questions and parse their responses, including a DAG that models the problem into symbolic expressions. Detailed prompts and parsing are demonstrated in Appendix F. Each prediction is then compared to the ground-truth using two metrics:

- **String Match:** A response is marked correct only if it matches the ground-truth expression exactly (after normalizing).
- **Symbolic (Ours):** A response is considered correct if it is derivable from the ground-truth using valid applications of do-calculus and probability rules.

Alternative metrics are discussed in Appendix G. We consider a prediction correct if the symbolic verifier finds a valid derivation within a depth limit of 20 steps, using our breadth-first search algorithm from Algorithm 1.

Results As shown in Table 1, our symbolic method identifies more correct answers than string match, raising the recall across all models and improving the F1 score accordingly. Our method is more useful when models such as `Llama3.1-8b` and `Mistral-7B` output an alternative form of the correct use. This improvement highlights an important phenomenon: many model responses are *causally correct* but fail naive evaluation due to superficial differences in formatting, variable order, or phrasing. The running time of verifying through BFS is minimal (milliseconds).

Our symbolic verifier recovers this missing accuracy by judging expressions based on their semantic content, not their surface form. It enables a more faithful and rigorous assessment of causal reasoning in LLMs, ensuring that models receive credit for valid reasoning even when their output does not match the reference verbatim.

Our symbolic verifier offers the largest improvements over string match for mid-performing models such as `Mistral-7B` and `Llama3.1-8B`, where syntactically different but semantically correct answers are common. In contrast, high-performing models like `Llama3.1-Instruct` already align well with ground-truth formats, so the relative gain over string match is smaller (e.g., F1 improves from 0.93 to 0.94). This suggests that the benefit of symbolic evaluation is most pronounced when models exhibit partial causal understanding but struggle with precise formalization. We identified several common patterns where symbolic verification offers substantial advantages:

Intervention with conditioning: Our system validates equivalence between expressions like $P(Y \mid \text{do}(X), Z = z)$ and $P(Y \mid \text{do}(X), Z)$ by correctly handling instantiated versus symbolic values.

Rule-based transformations: Our system correctly identifies that $P(Y \mid \text{do}(X), Z)$ can be transformed into $P(Y \mid X, Z)$ in DAGs where Z d-separates Y from incoming edges of X. This conversion from interventional to observational queries represented the majority of all verified equivalences. Note that this is important since the ground-truth of CLadder is in observational queries.

Multi-step proofs: For more complex cases, our verifier successfully applied sequential rules. These accounted fewer of verified equivalences but are representative of some of the most challenging verification scenarios.

4.3 Improving LLMs with Symbolic Feedback

Beyond evaluation, the proposed DoVerifier enables structured feedback to guide LLMs toward correct causal reasoning without relying on ground truth expressions. It has been shown that symbolic feedback loops (e.g., using SMT solvers in math or logic) have been shown to improve LLM output accuracy by providing formal, structured corrections (Hong et al., 2025; Murphy et al., 2024). Instead of using a reference answer as an oracle, our system leverages the causal graph structure and independence relationships to provide principled guidance.

Formal Description Given a causal graph $G = (V, E)$ (which may be LLM generated), an LLM generated expression $E_{\text{LLM}} = P(Y \mid \text{do}(X_1), \ldots, \text{do}(X_k), Z_1, \ldots, Z_m)$, and **no access to the ground truth** E_{target}. Our goal is to compute a revised expression E'_{LLM} that is causally more valid (i.e., more likely to match E_{target}) using structural reasoning over G.

We do so by partitioning the conditioning set of E_{LLM} into intervention variables $\mathbf{X}_{\text{do}}$ and $\mathbf{Z}_{\text{obs}}$:

$$\mathbf{X}_{\text{do}} = \{X_1, \ldots, X_k\} \quad \mathbf{Z}_{\text{obs}} = \{Z_1, \ldots, Z_m\}$$

Then, for each variable $Z \in \mathbf{Z}_{\text{obs}}$, we test:

- **Mediator Detection:** If Z lies on a directed path from some ancestor $A \in \mathbf{Z}_{\text{obs}} \cup \mathbf{X}_{\text{do}}$ to outcome Y:

$$A \to \cdots \to Z \to \cdots \to Y$$

  Then, Z is a mediator, so we write a prompt to avoid conditioning on Z, as doing so may block part of the causal pathway and lead to underestimation of the effect.

- **Treatment Confounding:** If $Z \in \mathbf{Z}_{\text{obs}}$ is a *common cause* of both a treatment variable $X \in \mathbf{X}_{\text{do}}$ and the outcome Y, i.e., $Z \to X$ and $Z \to Y$, then Z is a *confounder*. In such cases, we suggest replacing Z with $\text{do}(Z)$ when feasible, as intervening on Z may help eliminate confounding bias—particularly when front-door adjustment is applicable.

- **d-Separation Violation:** Let $\mathbf{W} = \mathbf{Z}_{\text{obs}} \setminus \{Z\} \cup \mathbf{X}_{\text{do}}$; if $X \not\perp Y \mid \mathbf{W}$, then we suggest conditioning on Z may bias the expression as it is not independent of Y given other variables $\mathbf{W}$.

Results Across all evaluated models, this feedback loop led to substantial gains in causal correctness. In our experiments, a significant portion of initially incorrect responses were corrected following a single feedback iteration. Table 2 shows the improvement of LLM performance using our feedback loop. We find that the effectiveness of symbolic feedback depends heavily on the type of error in the original expression. For example, when the model incorrectly uses $P(Y \mid X)$ instead of $P(Y \mid \text{do}(X))$, feedback guided by d-separation and rule-based reasoning often corrects the mistake. In contrast, if the model hallucinates an irrelevant variable or misrepresents the structure of the DAG itself, our framework is less effective since the symbolic transformations cannot fix structurally flawed inputs.

5 Discussions

This work formalizes the task of verifying causal correctness in language model outputs as a symbolic inference problem. The primary objective of the study is the derivation graph $S(E_{\text{init}})$ induced by the application of a finite rule set $\mathcal{R}$ (comprising do-calculus and probability transformations) to an initial causal expressions.

Metric	Llama3.1-8b			Mistral-7B			Llama-3.1-8B-Instruct			Gemma-7b-it		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Before Feedback	1.0	0.73	0.85	1.0	0.94	0.98	1.0	0.90	0.94	1.0	0.84	0.91
After Feedback	1.0	**0.93**	**0.97**	1.0	**0.99**	**0.99**	1.0	**0.98**	**0.99**	1.0	**0.87**	**0.93**

Table 2: Feedback improves LLM causal validity. Precision (P), Recall (R), and F1 scores before and after applying feedback derived from our verifier. The verifier enables models to revise incorrect expressions based solely on causal graph structure, boosting recall and overall correctness.

Semantic Equivalence as Proof-Theoretic Reachability We define semantic equivalence with respect to a causal graph G as the symmetric closure of the derivability relation:

$$E_1 \equiv_G E_2 \iff (E_1 \vdash_G E_2 \wedge E_2 \vdash_G E_1) \quad (9)$$

This defines a family of equivalence classes $[E]_{\equiv_G} \subset \mathcal{L}_{\text{causal}}$, where each class represents all expressions that are equivalent iff they encode the same interventional distribution in all causal models consistent with G. Empirically, we observe that LLM-generated outputs frequently fall into these equivalence classes without being string-identical to reference answers. For instance, expressions like $P(Y \mid X, Z)$ and $P(Y \mid \text{do}(X), Z)$ are lexically distinct but often semantically equivalent, conditional on specific d-separation statements. Our symbolic verifier resolves this not via heuristics, but by computing membership in the equivalence class through derivation.

Symbolic Feedback Works Because of Local Equivalence Neighborhoods In the presence of an incorrect LLM output E_{LLM}, our framework enables symbolic feedback by computing a correction E' such that

$$E' \in \text{Closure}_{\mathcal{R}}(E_{\text{LLM}}) \cap [E^*]_{\equiv_G} \quad (10)$$

where E^* is the latent correct expression (not known to the verifier). Operationally, this amounts to inverse proof search: finding a path from E_{LLM} to a semantically correct neighbor. Formally, the feedback procedure solves the following optimization:

$$\min_{E'} \{\text{cost}(E') \mid E_{\text{LLM}} \vdash_G E' \wedge E' \equiv_G E^*\} \quad (11)$$

In Section 4.3, we showed that providing feedback derived from symbolic analysis leads to improvements in model outputs. This supports the hypothesis that modern LLMs operate near locally correct regions of $\mathcal{L}_{\text{causal}}$, but lack explicit guarantees of logical closure (Wei et al., 2023; Zhou et al., 2023).

Failure Types Align with Non-derivability The most common model failures (e.g., using $P(Y \mid X)$ when X is a collider, or omitting keyfounders) correspond to derivations that fail d-separation conditions. For instance, symbolic proof fails when:

$$(Y \not\perp Z \mid X)_{G_{\overline{X}}} \implies$$
$$P(Y \mid X, Z) \not\equiv_G P(Y \mid \text{do}(X), Z) \quad (12)$$

These cases, which account for a significant portion of the errors in the models, are not just empirically incorrect but provably invalid under our formal system. This illustrates how symbolic reasoning captures not only surface alignment but deep structural correctness.

6 Conclusion

We introduced a formal verification framework, DoVerifier, for evaluating the causal validity of LLM-generated expressions. By modeling causal correctness as a symbolic derivation problem under do-calculus and probability rules, our system provides a principled alternative to string-based metrics for causal QA tasks. Through experiments on synthetic derivations and benchmarks like CLadder, we demonstrate that our method can detect subtle reasoning failures, recover semantically valid answers that are missed by naive metrics, and provide symbolic feedback that measurably improves model outputs.

Our results suggest that symbolic reasoning remains an essential component for trustworthy language models, especially in high-stakes domains where causal correctness matters. This work takes a step toward bridging the gap between natural language generation and formal reasoning systems by treating causal inference as a structured, verifiable process.

Limitations

While promising, our approach has several limitations and opens up for future work. On the one hand, the space of valid derivations can grow rapidly with the number of variables and the depth of allowed transformations. Although we employ optimizations like expression normalization and memoization, our breadth-first search remains computationally expensive in dense or deep DAGs. Future work could explore neural-guided proof search or approximate symbolic methods. On the other hand, regarding the feedback mechanism, the current feedback module improves the causal validity of model outputs using only the predicted DAG and the initial expression. It does not incorporate the original natural language question. As a result, the revised expression may be causally correct under the graph, but not necessarily faithful to the question intent. In practice, we observe that most LLM errors stem from misapplying causal semantics rather than misreading the question, but integrating question-aware feedback remains a valuable direction for future work.

Ethical Considerations

This work focuses on the formal verification of causal expressions generated by large language models (LLMs), with the goal of improving their semantic correctness and reliability in reasoning tasks. Our proposed framework does not involve human subject data, personally identifiable information, or real-world deployment in high-stakes settings such as healthcare or public policy. However, we acknowledge that causal claims can influence decision-making in sensitive domains. As such, we emphasize that symbolic correctness under do-calculus does not guarantee practical validity unless the underlying causal graph is itself accurate and contextually appropriate.

Our framework is designed for evaluation and diagnostic purposes, not for automating causal decisions. We caution against interpreting verified expressions as endorsements of correctness in real-world applications without domain expertise. To avoid misuse, we release our tools with clear disclaimers that they are intended for research and educational purposes.

References

AlphaProof and AlphaGeometry teams. 2024. Ai achieves silver-medal standard solving international mathematical olympiad problems. *https://deepmind.google/discover/blog/ ai-solves-imo-problems-at-silver-medal-level/*.

Alexander Bondarenko, Magdalena Wolska, Stefan Heindorf, Lukas Blübaum, Axel-Cyrille Ngonga Ngomo, Benno Stein, Pavel Braslavski, Matthias Hagen, and Martin Potthast. 2022. CausalQA: A benchmark for causal question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3296–3308, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Sirui Chen, Bo Peng, Meiqi Chen, Ruiqi Wang, Mengying Xu, Xingyu Zeng, Rui Zhao, Shengjie Zhao, Yu Qiao, and Chaochao Lu. 2024. Causal evaluation of language models. *Preprint*, arXiv:2405.00622.

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. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Leonardo de Moura and Sebastian Ullrich. 2021. The lean 4 theorem prover and programming language. In *Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings*, volume 12699 of *Lecture Notes in Computer Science*, pages 625–635. Springer.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *Preprint*, arXiv:1702.08608.

Jingxuan Fan, Sarah Martinson, Erik Y. Wang, Kaylie Hausknecht, Jonah Brenner, Danxian Liu, Nianli Peng, Corey Wang, and Michael Brenner. 2024. HARDMATH: A benchmark dataset for challenging problems in applied mathematics. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2025. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *The Thirteenth International Conference on Learning Representations*.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, Olli Järviniemi, Matthew Barnett, Robert Sandler, Matej Vrzala, Jaime Sevilla, Qiuyu Ren, Elizabeth Pratt, Lionel Levine, Grant Barkley, Natalie Stewart, Bogdan Grechuk, Tetiana Grechuk, Shreepranav Varma Enugandla, and Mark

Wildon. 2024. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *Preprint*, arXiv:2411.04872.

Aaron Grattafiori et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Sungee Hong, Zhengling Qi, and Raymond K. W. Wong. 2025. Distributional off-policy evaluation with bellman residual minimization. *Preprint*, arXiv:2402.01900.

Taojun Hu and Xiao-Hua Zhou. 2024. Unveiling llm evaluation focused on metrics: Challenges and solutions. *Preprint*, arXiv:2404.09135.

Albert Q. Jiang et al. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

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. 2023. Cladder: Assessing causal reasoning in language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 31038–31065. Curran Associates, Inc.

Nitish Joshi, Abulhair Saparov, Yixin Wang, and He He. 2024. LLMs are prone to fallacies in causal inference. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10553–10569, Miami, Florida, USA. Association for Computational Linguistics.

Zenan Li, Yifan Wu, Zhaoyu Li, Xinming Wei, Xian Zhang, Fan Yang, and Xiaoxing Ma. 2024. Autoformalize mathematical statements by symbolic equivalence and semantic consistency. *Preprint*, arXiv:2410.20936.

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. 2025. Goedel-prover: A frontier model for open-source automated theorem proving. *CoRR*, abs/2502.07640.

Jianqiao Lu, Yingjia Wan, Yinya Huang, Jing Xiong, Zhengying Liu, and Zhijiang Guo. 2024. Formalalign: Automated alignment evaluation for autoformalization. *CoRR*, abs/2410.10135.

Jing Ma. 2025. Causal inference with large language model: A survey. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5886–5898, Albuquerque, New Mexico. Association for Computational Linguistics.

Logan Murphy, Kaiyu Yang, Jialiang Sun, Zhaoyu Li, Anima Anandkumar, and Xujie Si. 2024. Autoformalizing euclidean geometry. *Preprint*, arXiv:2405.17216.

Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. 2002. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer.

OpenAI. 2024. Hello gpt-4o. Accessed: 2025-04-30.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

JUDEA PEARL. 1995. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688.

Judea Pearl. 2012. The do-calculus revisited. *Preprint*, arXiv:1210.4852.

ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, et al. 2025. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*.

Ivaxi Sheth, Bahare Fatemi, and Mario Fritz. 2025. CausalGraph2LLM: Evaluating LLMs for causal queries. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2076–2098, Albuquerque, New Mexico. Association for Computational Linguistics.

Ilya Shpitser and Judea Pearl. 2008. Complete identification methods for the causal hierarchy. *Journal of Machine Learning Research*, 9(64):1941–1979.

Gemma Team. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Gemma Team et al. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Santtu Tikka, Antti Hyttinen, and Juha Karvanen. 2021. Causal effect identification from multiple incomplete data sources: A general search-based approach. *Journal of Statistical Software*, 99(5):1–40.

Haiming Wang, Huajian Xin, Chuanyang Zheng, Zhengying Liu, Qingxing Cao, Yinya Huang, Jing Xiong, Han Shi, Enze Xie, Jian Yin, Zhenguo Li, and Xiaodan Liang. 2024. LEGO-prover: Neural theorem proving with growing libraries. In *The Twelfth International Conference on Learning Representations*.

Zeyu Wang. 2024. Causalbench: A comprehensive benchmark for evaluating causal reasoning capabilities of large language models. In *Causality and Large Models @NeurIPS 2024*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Huajian Xin, Luming Li, Xiaoran Jin, Jacques Fleuriot, and Wenda Li. 2025. Ape-bench i: Towards file-level automated proof engineering of formal math libraries. *arXiv preprint arXiv:2504.19110*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *Preprint*, arXiv:1904.09675.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. *Preprint*, arXiv:2205.10625.

A Desired Properties of a Good Verifier

A central question in the design of verifiers for symbolic causal reasoning is: what kinds of differences between derivations should not affect the evaluation? In other words, what transformations should a good evaluator be invariant to. In this section, we formalize the invariance and sensitivity properties that an ideal evaluator should satisfy. These properties are motivated both by formal semantics and by practical considerations in modeling causal reasoning.

Given an initial expression ϕ_0, a target expression ϕ^*, and a derivation sequence $\mathcal{D} = (\phi_0, \phi_1, \ldots, \phi_k = \phi^*)$, the evaluator should assign a score $s(\mathcal{D}) \in \mathbb{R}$ that reflects the logical correctness, minimality, and interpretability of the derivation.

Definition (Syntactic Equivalence). Let ϕ and ϕ' be probability expressions. We write $\phi \equiv_{\text{syn}} \phi'$ if they differ only by a syntactic permutation that preserves semantic content, such as reordering terms in a conditioning set:

$$P(Y \mid X, Z) \equiv_{\text{syn}} P(Y \mid Z, X) \tag{13}$$

Desideratum 1 (Syntactic Invariance). Let $\mathcal{D}$ be a derivation and $\mathcal{D}'$ a derivation obtained by a sequence of syntactic equivalences to the intermediate steps. Then:

$$s(\mathcal{D}) = s(\mathcal{D}') \tag{14}$$

Definition (α-Renaming). Let ϕ contain a variable V that does not appear free in other parts of the expression. Let ϕ' be the result of replacing V by V', where V' is a fresh variable name. Then $\phi \equiv_\alpha \phi'$.

Desideratum 2 (α-Renaming Invariance). The evaluator must satisfy

$$s(\mathcal{D}) = s(\mathcal{D}') \quad \text{if each } \phi_i' \equiv_\alpha \phi_i \text{ for all } i \tag{15}$$

Definition (Well-Typed Step). A step $\phi_i \rightarrow \phi_{i+1}$ using do-calculus Rule $r \in \{\text{Rule1}, \text{Rule2}, \text{Rule3}\}$ is valid if an only if the required graphical conditional independence is entailed by DAG G associated with the problem.

Desideratum 3 (Rule Sensitivity). If $\mathcal{D}$ and $\mathcal{D}'$ differ only in that $\mathcal{D}'$ includes a rule application r that violates the required independence, then:

$$s(\mathcal{D}') < s(\mathcal{D}) \tag{16}$$

This ensures the evaluator penalizes logically invalid or unsound reasoning.

Definition (Commutativity of Independent Steps). Let $\phi_i \rightarrow \phi_{i+1} \rightarrow \phi_{i+2}$ be two derivation steps, each applying a rule to a disjoint subformula of the expression. If $\mathcal{D}_1$ and $\mathcal{D}_2$ are derivations that only differ in the order of these two steps, then they are commutative.

Desideratum 4 (Step Order Invariance). We want $s(\mathcal{D}_1) = s(\mathcal{D}_2)$ if $\mathcal{D}_1, \mathcal{D}_2$ are commutative of independent steps to ensure the evaluator does not privilege arbitrary ordering of logically independent rule applications.

Definition (Derivational Equivalence). Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be distinct derivations from ϕ_0 to ϕ^*, where each step in both sequences is valid, though possibly differing in the choice or order of applied rules.

Desideratum 5 (Robustness to Valid Alternatives). The evaluator should satisfy $\forall \varepsilon > 0$:

$$|s(\mathcal{D}_1 - s(\mathcal{D}_2))| \leq \varepsilon \tag{17}$$

This encourages diversity in valid derivations without heavily penalizing alternative but correct reasoning paths.

B Implementation Details of DoVerifier

Our implementation converts abstract causal expressions into concrete computational objects that can be manipulated through rule applications. The core components are implemented as follows:

Expression Representation We represent causal expressions using a symbolic framework built on SymPy[3]. Each causal probability expression $P(Y \mid \text{do}(X), Z)$ is represented as a `CausalProbability` object with an outcome variable and a list of conditioning factors, which may include both observational variables and interventional variables (wrapped in `Do` objects). This representation allows for:

- Unique identification of expressions through consistent string conversion
- Distinction between interventional and observational variables
- Manipulation of expressions through rule applications

[3] https://www.sympy.org

Causal Graph Representation Causal graphs are represented using NetworkX[4] directed graphs, where nodes correspond to variables and edges represent causal relationships. For each rule application, we create modified graph structures according to the do-calculus definitions:

- For Rule 1, we remove incoming edges to intervention variables using $G_{\overline{X}}$
- For Rule 2, we remove both incoming edges to primary interventions and outgoing edges from secondary interventions using $G_{\overline{X}\underline{Z}}$
- For Rule 3, we perform the appropriate graph modifications for $G_{\overline{XZ(W)}}$ as specified by Pearl

D-separation Testing To determine rule applicability, we implement d-separation tests using NetworkX's built-in `is_d_separator` function. For each potential rule application, we:

1. Create the appropriate modified graph based on the rule
2. Identify the variables that need to be tested for conditional independence
3. Perform the d-separation test with the appropriate conditioning set
4. Apply the rule only if the independence condition is satisfied

For example, when applying Rule 1 to remove an observation Z from $P(Y \mid \text{do}(X), Z)$, we test whether Y and Z are d-separated given X in the graph $G_{\overline{X}}$.

Search Algorithm Optimization To make the breadth-first search efficient, we implement several optimizations:

- **Expression normalization:** We convert expressions to canonical string representations with consistent ordering and whitespace removal.
- **Memoization:** We cache the results of d-separation tests to avoid redundant graph operations.
- **Early termination:** We immediately return a proof path when the target expression is found.
- **Visited set tracking:** We maintain a set of already-visited expressions to avoid cycles and redundant exploration.

[4]https://networkx.org/

Handling Incomplete Knowledge A key innovation in our implementation is the ability to work with incomplete causal knowledge. When the full DAG structure is unknown, our system can:

- Work with explicitly provided independence pairs between variables
- Infer independence relationships from partial graph information
- Explore potential equivalences under different assumptions

Scope of Verification While our implementation includes representations for both probability distributions (P) and expectations (E), our current verification framework focuses on causal expressions involving probabilities. This focus aligns with Pearl's do-calculus, which was formulated for probability distributions. The identification of causal effects fundamentally involves transforming interventional probabilities into expressions based on observed data.

The framework can be extended to handle expectations directly, as we have implemented the necessary data structures and fundamental operations for expectation expressions. However, since expectations are functionals of probability distributions, verifying equivalence at the probability level is sufficient for most practical causal inference tasks. Once the correct probability expression is identified, expectations and other functionals can be derived through standard statistical methods.

C Proof of Theorem 3.2

We formally prove the soundness and completeness of our verification framework by modeling it as a symbolic derivation system over a finite-branching graph induced by transformation rules.

Proposition C.1 (State Space as Derivation Graph)**.** *Let G be a causal DAG and let E_{init} be a valid causal expression. Then the set of all expressions derivable from E_{init} using do-calculus and probability rules forms a well-defined directed graph $S(E_{init})$ over the language $\mathcal{L}_{causal}$, where:*

- *Nodes correspond to normalized symbolic expressions over G.*

- *Directed edges correspond to valid applications of transformation rules.*

Proof. Each expression in $\mathcal{L}_{\text{causal}}$ can be represented as a symbolic term $P(Y \mid \mathbf{Z})$, where $\mathbf{Z}$ contains a mix of observational and interventional variables. Given a finite set of transformation rules $\mathcal{R}$ (including the rules of do-calculus, Bayes' rule, chain rule, etc.), we define an edge from $E \to E'$ if E' can be obtained from E via one rule in $\mathcal{R}$ under G.

Since expressions are finitely representable (e.g., via normalized strings) and rules are deterministic, $S(E_{\text{init}})$ is a well-defined labeled transition graph over expressions reachable from E_{init}. ■

Proposition C.2 (Finiteness of Branching Factor)**.** *For any node E in the derivation graph $S(E_{init})$, the number of distinct expressions reachable via a single transformation is finite.*

Proof. Let $n = |\mathcal{V}|$ be the number of nodes in the DAG G.

For each rule $r \in \mathcal{R}$:

- **Rule 1** can be applied at most once for each Z in the conditioning set $\mathbf{Z}$, giving at most n applications.
- **Rule 2** and **Rule 3** apply to intervention variables, also bounded by n.
- **Bayes' Rule** applies to pairs of variables (X, Y), yielding at most $\mathcal{O}(n^2)$ applications.
- **Chain Rule** considers permutations over subsets of $\mathcal{V}$, giving at most $\mathcal{O}(n!)$ possibilities.
- **Law of Total Probability** can be applied per mediator variable, also bounded by n.

Thus, each node E in $S(E_{\text{init}})$ has a finite out-degree, i.e., the branching factor is finite. ■

Proposition C.3 (Completeness of BFS)**.** *Let $E_{init}, E_{target} \in \mathcal{L}_{causal}$ be expressions defined over the same causal graph G. If there exists a derivation sequence*

$$E_{init} \to E_{target}$$

of length at most d under rules $\mathcal{R}$, then breadth-first search with depth limit d will find such a sequence. Moreover, if no such sequence exists within d steps, BFS will terminate after enumerating all reachable expressions within that bound.

Proof. We model derivations as paths in the state graph $S(E_{\text{init}})$. Since the graph has a finite branching factor (Theorem C.2) and rule applications are deterministic, breadth-first search explores this graph in increasing depth order.

For any reachable expression E_{target} such that $E_{\text{init}} \vdash_G E_{\text{target}}$ within d steps, BFS is guaranteed to enumerate it after at most $\mathcal{O}(b^d)$ steps, where b is the maximum branching factor. Furthermore, BFS finds the shortest such path in terms of rule applications, since it explores all paths of length k before those of length $k+1$. ■

D Practical Considerations

Fact D.1 (Complexity)**.** *The time complexity of BFS is $\mathcal{O}(b^d)$ where b is the maximum branching factor and d is the depth limit.*

While theoretically sound, practical implementations must consider several optimizations:

1. **Expression normalization** to avoid revisiting equivalent states (e.g., removing redundant conditions, standardizing variable order)
2. **Efficient d-separation testing** for determining rule applicability
3. **Memoization** of independence tests to avoid redundant graph operations
4. **Strategic ordering of rule applications** to potentially find solutions faster
5. **Bidirectional search** from both E_{init} and E_{target} to reduce the effective search depth

These optimizations preserve the theoretical guarantees while making the approach computationally feasible for practical use in evaluating causal reasoning in language models.

E Data Samples of Synthetic Data

To support the evaluation of causal inference methods, we construct synthetic datasets using directed acyclic graphs (DAGs) that encode assumed causal relationships among variables. Each DAG consists of nodes representing variables and directed edges representing direct causal influences. These graphs serve as the basis for simulating both observational and interventional data.

The data samples are designed to validate derivations using do-calculus. Each example contains:

- A **DAG** representing the underlying relationships.
- A pair of probability expressions (E_a, E_b) where E_a is an interventional expression involving do-operators and E_b is an equivalent or simplified observational expression.
- A proof showing the sequence of do-calculus rules (Rule 1, Rule 2, Rule 3) applied to reduce E_a to E_b. These synthetic samples are not drawn from real-world distributions, but they adhere strictly to the independence constraints implied by the DAGs, ensuring the theoretical correctness of all derivations.

F Prompt Examples

To evaluate and guide language model performance on causal reasoning tasks, we designed a two-shot prompt that consists of: A set of instructions, two fully worked examples, a new query prompt for the model to solve in the same format.

```
## Instructions:
1. For each problem, identify the correct
    ↪ expression that represents the query
2. Draw the graphical representation as a
    ↪ text description of edges
3. Show your mathematical reasoning step by
    ↪ step
4. Provide a final yes/no answer
5. Keep your response concise and focused on
    ↪  the solution

## Examples:

Example 1:
Prompt: Imagine a self-contained,
    ↪ hypothetical world with only the
    ↪ following conditions, and without any
    ↪  unmentioned factors or causal
    ↪ relationships: Poverty has a direct
    ↪ effect on liking spicy food and
    ↪ cholera. Water company has a direct
    ↪ effect on liking spicy food. Liking
    ↪ spicy food has a direct effect on
    ↪ cholera. Poverty is unobserved. The
    ↪ overall probability of liking spicy
    ↪ food is 81%. The probability of not
    ↪ liking spicy food and cholera
    ↪ contraction is 13%. The probability
    ↪ of liking spicy food and cholera
    ↪ contraction is 17%. Is the chance of
    ↪ cholera contraction larger when
    ↪ observing liking spicy food?
Let V2 = water company; V1 = poverty; X =
    ↪ liking spicy food; Y = cholera

Expression: P(Y | X)
Graphical Representation: V1->X,V2->X,V1->Y,
    ↪ X->Y
Reasoning: P(X = 1, Y = 1)/P(X = 1) - P(X =
    ↪ 0, Y = 1)/P(X = 0)
P(X=1) = 0.81
P(Y=1, X=0) = 0.13
P(Y=1, X=1) = 0.17
0.17/0.81 - 0.13/0.19 = -0.44
-0.44 < 0
Final Answer: No

Example 2:
Prompt: Imagine a self-contained,
    ↪ hypothetical world with only the
    ↪ following conditions, and without any
    ↪  unmentioned factors or causal
    ↪ relationships: Poverty has a direct
    ↪ effect on liking spicy food and
    ↪ cholera. Water company has a direct
    ↪ effect on liking spicy food. Liking
    ↪ spicy food has a direct effect on
    ↪ cholera. Poverty is unobserved. For
    ↪ people served by a local water
    ↪ company, the probability of cholera
    ↪ contraction is 64%. For people served
    ↪  by a global water company, the
    ↪ probability of cholera contraction is
    ↪  66%. For people served by a local
    ↪ water company, the probability of
    ↪ liking spicy food is 50%. For people
    ↪ served by a global water company, the
    ↪  probability of liking spicy food is
    ↪ 45%. Will liking spicy food decrease
    ↪ the chance of cholera contraction?
Let V2 = water company; V1 = poverty; X =
    ↪ liking spicy food; Y = cholera.

Expression: E[Y | do(X = 1)] - E[Y | do(X =
    ↪ 0)]
Graphical Representation: V1->X,V2->X,V1->Y,
    ↪ X->Y
Reasoning: E[Y | do(X = 1)] - E[Y | do(X =
    ↪ 0)]
[P(Y=1|V2=1)-P(Y=1|V2=0)]/[P(X=1|V2=1)-P(X
    ↪ =1|V2=0)]
P(Y=1 | V2=0) = 0.64
P(Y=1 | V2=1) = 0.66
P(X=1 | V2=0) = 0.50
P(X=1 | V2=1) = 0.45
(0.66 - 0.64) / (0.45 - 0.50) = -0.39
-0.39 < 0
Final Answer: Yes

## Your Task:
Solve the following problem using the format
    ↪  above. Begin your response with "
    ↪ Solution:" and provide only the
    ↪ expression, graphical representation,
    ↪  reasoning, and final answer.
Prompt: {description}
```

G Alternative Metrics

Evaluation of causal expression generation has often relied on surface-level metrics such as exact string match, BLEU score, BERTscore, and token-level F1.

BLEU and Token-level F1 Fails for Causal Evaluation BLEU computes precision over n-grams between a candidate and reference string. In causal

Model	BLEU	Token-level F1
Llama-3.1-8B-Instruct (Grattafiori et al., 2024)	0.46	0.70
Mistral-7B-v0.1 (Jiang et al., 2023)	0.33	0.58
Llama-3.1-8B (Grattafiori et al., 2024)	0.36	0.57
Gemma-7b-it (Team et al., 2024)	0.19	0.55

Table 3: Average BLEU and token-level F1 scores for each model evaluated on CLadder.

<table>
<tr><th>LLM Output</th><th>Formal Label</th><th>Correct?</th><th>BERTScore F1</th></tr>
<tr><td><code>P(Y | V1)</code></td><td><code>P(Y | X)</code></td><td>No</td><td>0.91</td></tr>
<tr><td><code>P(Y)</code></td><td><code>P(Y | X)</code></td><td>No</td><td>0.91</td></tr>
</table>

Table 4: Incorrect model outputs with high BERTScore. While these expressions differ from the gold standard, BERTScore assigns high similarity, demonstrating its over-generosity in causal evaluation.

reasoning, it suffers from

Small expression length bias: Causal expressions are often short; hence, BLEU becomes unstable when evaluating < 10 token strings since higher-order n-grams vanish.

Syntactic Fragility: Expressions that are semantically equivalent but have different variable order get penalized.

Non-semantic penalties: BLEU may still reward inclusion of irrelevant variables if they overlap with the gold string, even if the overall expression is wrong.

Token-level F1 computes overlap between tokens, treating the expression as a bag of symbols. It however, still leads to multiple failure cases:

Ignores structure role of variables: F1 cannot distinguish $P(Y)$ from $P(Y \mid X)$ or $P(Y \mid \mathrm{do}(X))$. They call share some subset of overlapping tokens and will inflate the recall.

No notion of well-formedness: Syntactically expressions such as $P(X\ Y)$ or $Y \mid P(X)$ might have high F1 if they reuse common symbols despite being invalid.

No semantics: Conditioning vs intervention is completely ignored, a model can be rewarded for guessing the right letters, not the right logic.

Table 3 shows the average BLEU and token-level F1 score for each model evaluated on causal language tasks. We see that both BLEU and F1 lack a formal grounding in the semantics of causal inference. There is no transformation set $\mathcal{T}$ under which they define an equivalence class. In contrast, our symbolic verifier defines:

$$\phi_1 \equiv_G \phi_2 \iff \phi_1 \vdash_G \phi_2 \wedge \phi_2 \vdash_G \phi_1 \quad (18)$$

Thus, BLEU and F1 may disagree with formal correctness, and worse, may systematically overestimate the validity of incorrect outputs.

BERTScore Failure Cases BERTScore (Zhang et al., 2020) is a widely used metric that computes semantic similarity by aligning contextualized token embeddings from a pretrained BERT model. It is often promoted as a semantically aware alternative to BLEU. However, in the context of causal reasoning, BERTScore exhibits a distinct failure mode: it confuses lexical proximity for logical validity. Table 4 shows common failure cases where BERTscore assigns a high similarity score, even when they are not supposed to be equivalent expressions. Let $\phi_{\text{pred}}, \phi_{\text{gold}} \in \mathcal{L}_{\text{causal}}$ be causal expressions encoded as strings. BERTScore computes:

$$\text{BERTScore}(\phi_{\text{pred}}, \phi_{\text{gold}}) = \text{F1}_{\text{BERT}}(h_{\phi_{\text{pred}}}, h_{\phi_{\text{gold}}}) \quad (19)$$

where h_ϕ are contextual embeddings from a pretrained BERT model. However, the model has no knowledge of causal semantics, independence structures, or the syntax of do-calculus. Tokens like `P`, `(`, `)` are close in embedding space regardless of their role in the logical formula. This results in BERTScore assigning high similarity to expressions that are semantically disjoint under the causal graph. Unlike DoVerifier, BERTScore lacks a soundness guarantee

$$\text{BERTScore}(\phi_{\text{pred}}, \phi_{\text{gold}}) > 0.9 \not\Rightarrow \phi_{\text{pred}} \equiv_G \phi_{\text{gold}} \tag{20}$$

This could become dangerous in high-stakes contexts, where plausible-looking causal statements may lead to incorrect conclusions when evaluated with BERTScore.