Capturing Semantic Correctness for Causal Reasoning Evaluation via Symbolic Verification

Paul He[1,2,6*] **Yinya Huang**[3,5] **Mrinmaya Sachan**[3,5] **Zhijing Jin**[1,2,4]
[1]University of Toronto [2]Vector Institute [3]ETH Zürich
[4]Max Planck Institute for Intelligent Systems [5]ETH AI Center [6]Amazon Web Services
{hepaul, zjin}@cs.toronto.edu

Abstract

Large language models (LLMs) are increasingly applied to tasks involving causal reasoning. However, current benchmarks often rely on string matching or surface-level metrics that fail to assess whether a model's output is formally valid under causal semantics. We propose DoVerifier, a symbolic verification framework that checks whether LLM-generated causal expressions are derivable from a given causal graph using rules from do-calculus and probability theory. This allows us to recover correct answers that would otherwise be marked incorrect due to superficial differences. Evaluations on synthetic data and causal QA benchmarks show that DoVerifier more accurately captures semantic correctness than standard metrics, offering a more rigorous and informative way to evaluate LLMs on causal tasks.

1 Introduction

Causal reasoning enables humans to explain effects, predict interventions, and reason about counterfactuals. As large language models (LLMs) [OpenAI, 2024, Team, 2025, DeepSeek-AI, 2025] are increasingly applied in science, medicine, and policy, the ability to generate and verify causal claims is critical for trustworthy AI [Doshi-Velez and Kim, 2017]. An LLM that distinguishes correlation from causation could support tasks from experimental design to hypothesis generation.

Recent benchmarks such as CLadder [Jin et al., 2023] and CausalBench [Wang, 2024] evaluate LLMs on causal question answering. Yet their evaluation relies on surface-level metrics (e.g., exact match, BLEU, token-level F1, BERTScore), which capture surface-level string similarity rather than semantic correctness. As a result, logically valid causal expressions that differ syntactically may be penalized, while invalid ones may score highly.

This limitation reflects a broader gap: causal inference depends on causal semantics. The validity of $P(Y \mid (X))$ is determined not by its string form but by derivability from a causal graph using probability rules and do-calculus [Pearl, 1995]. Unlike mathematical reasoning tasks [Gao et al., 2025, Fan et al., 2024, Cobbe et al., 2021, Hendrycks et al., 2021], where correctness can often be checked numerically, causal reasoning rarely permits substitution into a full joint distribution.

We propose DoVerifier, a symbolic verification framework that checks whether LLM-generated causal expressions are derivable using do-calculus. This approach captures equivalences missed by string metrics and provides a principled basis for evaluating causal reasoning. Our experiments show that DoVerifier recovers semantically correct outputs overlooked by existing benchmarks, enabling more rigorous assessment of LLM's causal reasoning abilities.

*Work done prior joining AWS.

Accepted to the NeurIPS 2025 Workshop on CauScien: Uncovering Causality in Science.

2 Related Work

Evaluation of causal QA in LLMs has largely relied on surface similarity. Benchmarks such as CLadder [Jin et al., 2023] and CausalBench [Wang, 2024] probe associational, interventional, and counterfactual queries, but scoring is based on string overlap (e.g., BLEU [Papineni et al., 2002], token-level F1, BERTScore [Zhang et al., 2020]). Such metrics often misclassify outputs: logically equivalent expressions may be penalized, while semantically wrong answers can score high due to token overlap.

In causal inference, do-calculus [Pearl, 1995] and the ID algorithm [Shpitser and Pearl, 2008, Tikka et al., 2021] provide sound procedures for determining whether a causal effect is identifiable and, if so, deriving its estimand. One might suggest using ID to simplify each expression and then compare whether the simplified results are equal. However, this approach is limited to identifiable cases: if a query is unidentifiable, ID returns failure rather than a transformed expression. In contrast, DoVerifier is designed to reason symbolically about equivalence even when effects are unidentifiable but simplifiable through valid applications of do-calculus. For example,

$$E_1 = P(Y \mid \mathrm{do}(X), \mathrm{do}(W), Z) \quad \text{and} \quad E_2 = P(Y \mid \mathrm{do}(X), Z) \tag{1}$$

are unidentifiable under many graphs, yet DoVerifier can still establish their equivalence by recognizing that W is irrelevant given Z. Thus, our framework generalizes beyond identification to the broader task of verifying semantic equivalence between causal expressions. Recent work has aligned the answers with causal graphs using templates [Sheth et al., 2025], but without symbolic derivability.

Related efforts in mathematical reasoning also explore formal verification. Systems like Lean and Isabelle have been used to check proofs [Ren et al., 2025, Wang et al., 2024], while SMT-based approaches assess equivalence in geometry and logic [Murphy et al., 2024, Li et al., 2024]. We build on this paradigm but extend it to causal inference, where correctness depends on do-calculus derivability in a causal DAG.

3 DoVerifier: Causal Symbolic Verification Framework

3.1 Preliminaries

Let $G = (V, E)$ be a causal DAG over a finite set of observed variables V. We consider a symbolic language of causal expressions defined as follows.

Definition 3.1 (Causal Expression and Language)**.** *A* causal expression *is any term of the form*

$$P(Y \mid \mathbf{Z}) \quad \textit{or} \quad P(Y \mid \mathbf{Z}, do(\mathbf{X})), \tag{2}$$

where $Y, \mathbf{Z}, \mathbf{X} \subseteq V$ are disjoint variable sets and $do(\mathbf{X})$ denotes interventions on $\mathbf{X}$. The set of all well-formed expressions under G is denoted $\mathcal{L}_{causal}$. Two expressions $E_1, E_2 \in \mathcal{L}_{causal}$ are equivalent under *G, written $E_1 \equiv_G E_2$, if and only if each can be derived from the other through valid applications of do-calculus and probability rules consistent with G.*

We write $E_{\text{init}} \vdash_G E_{\text{target}}$ to denote that E_{target} is derivable from E_{init} via a finite sequence of rule applications, while respecting the conditional independencies encoded by G. Intuitively, $\vdash_G$ represents the *entailment* relation induced by the causal graph: if $E_{\text{init}} \vdash_G E_{\text{target}}$, then the two expressions are semantically consistent with the same underlying causal structure.

Problem Statement. Given a causal DAG G and two expressions $E_1, E_2 \in \mathcal{L}_{\text{causal}}$, our goal is to determine whether they are causally equivalent under G. Formally, we seek a verifier

$$\mathcal{F} : \mathcal{L}_{\text{causal}} \times \mathcal{L}_{\text{causal}} \times \mathcal{G} \to [0, 1], \tag{3}$$

such that $\mathcal{F}(E_1, E_2, G) = 1$ iff $E_1 \vdash_G E_2$. This problem formulation underpins DoVerifier, a system that symbolically checks causal equivalence by searching for valid derivations under the rules of do-calculus and probability theory. A summary of the desired properties of a sound verifier is provided in Appendix A.

Example. Consider $E_1 = P(F \mid \mathrm{do}(A), \mathrm{do}(B), C)$ and $E_2 = P(F \mid \mathrm{do}(B))$ under a graph where A only affects F through B. A string-based metric would reject equivalence, but $E_1 \vdash_G E_2$ holds by do-calculus.

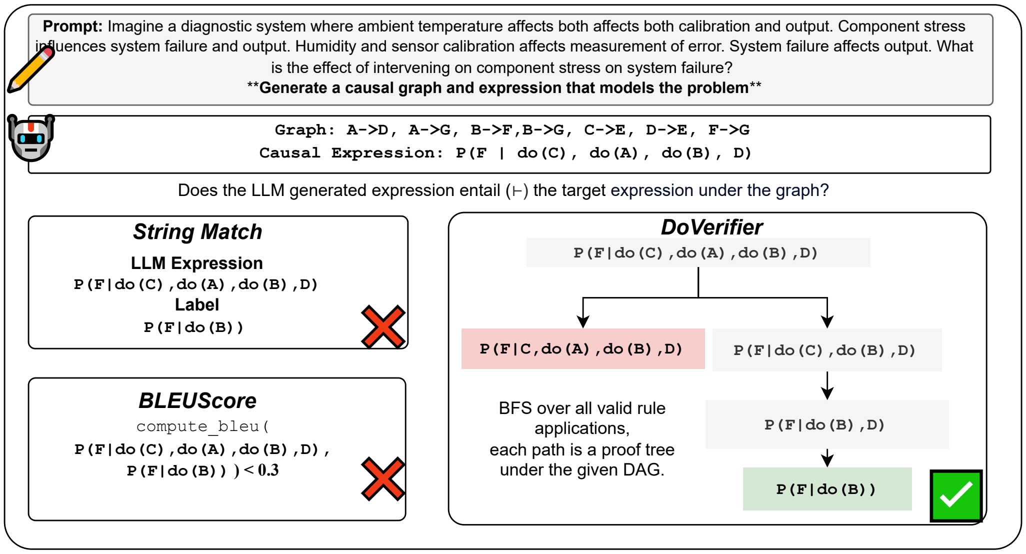

Figure 1: Overview of the DoVerifier pipeline. The LLM generates a causal graph G and an expression E_{pred}. The expression is parsed into symbolic form, and a breadth-first proof search applies the rule set $\mathcal{R}$ to derive new expressions. If the ground-truth expression E_{target} is reached, the verifier certifies their equivalence.

Do-calculus rules (sketch) Do-calculus provides three transformation rules governing insertion/deletion of observations and actions. These rules allow us to reduce interventions or exchange them with observations under appropriate d-separation conditions. We use them as the basis of symbolic proof search in DoVerifier. (Full formal statements are given in Appendix B.)

3.2 Method

Our method, DoVerifier, aims to verify whether an expression $E_{\text{pred}} \in \mathcal{L}_{\text{causal}}$ produced by the LLM is equivalent to a ground-truth expression $E_{\text{target}} \in \mathcal{L}_{\text{causal}}$ given a causal graph G. The graph G may be provided or generated by the LLM itself. Formally, we seek to decide whether

$$E_{\text{pred}} \equiv_G E_{\text{target}},$$

i.e., whether both expressions evaluate to the same interventional distribution under the causal model represented by G.

Proof Search. Starting from E_{pred}, DoVerifier performs a breadth-first search (BFS) over all expressions reachable by applying the rules in $\mathcal{R}$. In the resulting *derivation tree*, each node corresponds to a well-formed expression, and each directed edge corresponds to a valid rule application. A sequence of applications

$$E_{\text{pred}} \xrightarrow{R_{i_1}} E_1 \xrightarrow{R_{i_2}} \dots \xrightarrow{R_{i_k}} E_{\text{end}} \tag{4}$$

represents a single root-to-node path in this tree.

If $E_{\text{end}} = E_{\text{target}}$, the verifier declares equivalence; otherwise, the search continues until no new expressions can be generated under the given rules and depth limit. Because each rule in $\mathcal{R}$ corresponds to a sound transformation under the causal semantics of G, any successful derivation certifies a valid equivalence. Moreover, since BFS systematically explores all possible rule applications without repetition, the search is complete for a finite $\mathcal{R}$ and variable set. Formal proofs of these properties are provided in Appendix D.

Model	String Match	LLM-as-a-judge	BLEU	Token-F1	DoVerifier (Ours)
Llama3.1-8B	0.57	0.60	0.36	0.57	**0.73**
Mistral-7B	0.58	0.80	0.33	0.58	**0.94**
Llama3.1-8B-Instruct	0.88	0.66	0.46	0.70	**0.90**
Gemma-7B-it	0.80	0.58	0.19	0.55	**0.84**

Table 1: DoVerifier recovers more correct causal expressions than string match, LLM-as-a-judge, BLEU, or token-level F1 across four LLMs on CLadder. Our method identifies semantically valid expressions missed by surface-level metrics.

4 Experiments and Results

4.1 Synthetic Data Test

To test the verifier, and to show that existing metrics fail in cases where syntactically different expressions are the same semantically, 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. We provide the sampling procedure details in Appendix F. 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 syntactically.

4.2 LLM Causal Reasoning Evaluation

Next, we evaluate how well large language models (LLMs) perform on causal reasoning tasks and how our symbolic verifier (DoVerifier) improves their evaluation. Specifically, we ask: *Can our method recover correct answers that naive metrics miss?*

Setup. We evaluate models on the **CLadder** benchmark [Jin et al., 2023], a suite of causal questions grounded in known DAGs, each paired with a ground-truth causal expression. Four models are tested—`Llama-3.1-8B`, `Llama-3.1-8B-Instruct` [Grattafiori et al., 2024], `Mistral-7B` [Jiang et al., 2023], and `Gemma-7B-it` [Team et al., 2024]. For each question, the model output is parsed into a symbolic form and compared to the ground truth using multiple evaluation schemes. **String Match** counts a prediction correct only if it exactly matches the normalized reference. **LLM-as-a-judge** uses GPT-4o [OpenAI, 2024] to assess whether the predicted and reference expressions are equivalent given the DAG. **BLEU** and **Token-F1** serve as standard text-similarity baselines measuring n-gram or token overlap. Finally, **Symbolic (Ours)** deems an answer correct if it can be derived from the reference through valid applications of do-calculus and probability rules within 20 inference steps.

Results. As shown in table 1, DoVerifier consistently recovers additional correct answers across all models. Many LLM outputs are *causally correct* yet fail under string or lexical metrics due to minor differences in syntax, variable order, or formatting. Our verifier restores these missed cases by checking semantic equivalence rather than surface form, running efficiently in milliseconds.

High-performing models such as `Llama3.1-8B-Instruct` benefit less because their outputs already align with reference syntax, while mid-performing models (`Mistral-7B`, `Llama3.1-8B`) gain the most—showing that symbolic verification is especially valuable when models grasp causal logic but produce alternative formulations. Unlike LLM-as-a-judge, our method guarantees soundness[2], avoiding overinterpretation or inconsistency introduced by large evaluators.

Limitations of Alternative Metrics. Conventional similarity metrics such as BLEU, token-level F1, and BERTScore fail to capture causal correctness because they evaluate surface similarity rather than semantic validity. BLEU, which measures n-gram precision, is unstable for short expressions and penalizes harmless reorderings or equivalent reformulations, often rewarding spurious token overlaps instead of genuine equivalence. Token-level F1 performs slightly better but still ignores

[2]String match is sound but incomplete.

structure—expressions such as $P(Y)$, $P(Y \mid X)$, and $P(Y \mid \text{do}(X))$ share most tokens yet differ fundamentally in meaning.

BERTScore [Zhang et al., 2020] extends these metrics by comparing contextual embeddings of tokens, computed as

$$\text{BERTScore}(\phi_{\text{pred}}, \phi_{\text{gold}}) = \text{F1}_{\text{BERT}}(h_{\phi_{\text{pred}}}, h_{\phi_{\text{gold}}}), \tag{5}$$

where h_ϕ are contextualized embeddings from a pretrained language model. However, these embeddings contain no notion of causal semantics—tokens like `P`, `(`, or `do` are treated as similar regardless of their logical role. Consequently,

$$\text{BERTScore}(\phi_{\text{pred}}, \phi_{\text{gold}}) > 0.9 \not\Rightarrow \phi_{\text{pred}} \equiv_G \phi_{\text{gold}}. \tag{6}$$

In contrast, our verifier defines causal equivalence through derivability:

$$\phi_1 \equiv_G \phi_2 \iff \phi_1 \vdash_G \phi_2 \wedge \phi_2 \vdash_G \phi_1, \tag{7}$$

which grounds evaluation in the formal semantics of do-calculus. This guarantees both soundness and completeness with respect to the causal graph, providing a principled alternative to metrics that reward syntactic or embedding-level similarity without causal validity.

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.

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:

$$E_1 \equiv_G E_2 \iff (E_1 \vdash_G E_2 \wedge E_2 \vdash_G E_1) \tag{8}$$

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.

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 confounders) correspond to derivations that fail d-separation conditions. For instance, symbolic proof fails when:

$$(Y \not\perp\!\!\!\perp Z \mid X)_{G_{\overline{X}}} \implies P(Y \mid X, Z) \not\equiv_G P(Y \mid \text{do}(X), Z) \tag{9}$$

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 DoVerifier, a formal verification framework that evaluates the causal validity of LLM-generated expressions by modeling causal reasoning as a symbolic derivation task using do-calculus and probability rules. Our approach captures semantically equivalent causal expressions that are missed by standard metrics, improving recall on causal benchmarks, and enabling structured feedback to refine model outputs. These findings reveal a significant gap in current evaluations of causal reasoning and highlight the importance of symbolic verification for building reliable causal reasoning systems. By connecting natural language generation with formal inference, DoVerifier offers a principled step toward evaluating models on causal reasoning based on what they truly understand rather than how they phrase it.

References

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. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning, 2017. URL https://arxiv.org/abs/1702.08608.

Jingxuan Fan, Sarah Martinson, Erik Y. Wang, Kaylie Hausknecht, Jonah Brenner, Danxian Liu, Nianli Peng, Corey Wang, and Michael Brenner. HARDMATH: A benchmark dataset for challenging problems in applied mathematics. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024. URL https://openreview.net/forum?id=gt6prlTEGL.

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. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yaqPf0KAlN.

Aaron Grattafiori et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

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, 2021. URL https://arxiv.org/abs/2103.03874.

Albert Q. Jiang et al. Mistral 7b, 2023. URL https://arxiv.org/abs/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. Cladder: Assessing causal reasoning in language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 31038–31065. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/631bb9434d718ea309af82566347d607-Paper-Conference.pdf.

Zenan Li, Yifan Wu, Zhaoyu Li, Xinming Wei, Xian Zhang, Fan Yang, and Xiaoxing Ma. Autoformalize mathematical statements by symbolic equivalence and semantic consistency, 2024. URL https://arxiv.org/abs/2410.20936.

Logan Murphy, Kaiyu Yang, Jialiang Sun, Zhaoyu Li, Anima Anandkumar, and Xujie Si. Autoformalizing euclidean geometry, 2024. URL https://arxiv.org/abs/2405.17216.

OpenAI. Hello gpt-4o, May 2024. URL https://openai.com/index/hello-gpt-4o/. Accessed: 2025-04-30.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 12 1995. ISSN 0006-3444. doi: 10.1093/biomet/82.4.669. URL https://doi.org/10.1093/biomet/82.4.669.

Judea Pearl. The do-calculus revisited, 2012. URL https://arxiv.org/abs/1210.4852.

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

Ivaxi Sheth, Bahare Fatemi, and Mario Fritz. CausalGraph2LLM: Evaluating LLMs for causal queries. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2076–2098, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. URL https://aclanthology.org/2025.findings-naacl.110/.

Ilya Shpitser and Judea Pearl. Complete identification methods for the causal hierarchy. *Journal of Machine Learning Research*, 9(64):1941–1979, 2008. URL http://jmlr.org/papers/v9/shpitser08a.html.

Gemma Team. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Gemma Team et al. Gemma: Open models based on gemini research and technology, 2024. URL https://arxiv.org/abs/2403.08295.

Santtu Tikka, Antti Hyttinen, and Juha Karvanen. Causal effect identification from multiple incomplete data sources: A general search-based approach. *Journal of Statistical Software*, 99(5):1–40, 2021. doi: 10.18637/jss.v099.i05. URL https://www.jstatsoft.org/index.php/jss/article/view/v099i05.

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. LEGO-prover: Neural theorem proving with growing libraries. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3f5PALef5B.

Zeyu Wang. Causalbench: A comprehensive benchmark for evaluating causal reasoning capabilities of large language models. In *Causality and Large Models @NeurIPS 2024*, 2024. URL https://openreview.net/forum?id=kbmGbm2L1P.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020. URL https://arxiv.org/abs/1904.09675.

A Desired Properties of a Good Verifier

A central question in the design of verifiers for symbolic causal reasoning $\mathcal{F}$ 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:

$$P(Y \mid X, Z) \equiv_{\text{syn}} P(Y \mid Z, X) \tag{10}$$

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:

$$s(\mathcal{D}) = s(\mathcal{D}') \tag{11}$$

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

$$s(\mathcal{D}) = s(\mathcal{D}') \quad \text{if each } \phi_i' \equiv_\alpha \phi_i \text{ for all } i \tag{12}$$

Definition (Well-Typed Step). A step $\phi_i \rightarrow \phi_{i+1}$ using do-calculus Rule $r \in \{\text{Rule15}, \text{Rule16}, \text{Rule17}\}$ 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:

$$s(\mathcal{D}') < s(\mathcal{D}) \tag{13}$$

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 $\forall \varepsilon > 0$:

$$|s(\mathcal{D}_1) - s(\mathcal{D}_2)| \leq \varepsilon \tag{14}$$

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

B Do-Calculus

Unlike factual QA, causal evaluation is not always numeric: we cannot simply plug in values to verify an answer. Instead, we must determine whether an expression like $P(Y \mid (X))$ follows logically from a known graph structure.

To determine whether $E_1 \vdash_G E_2$ holds, we rely on the rules of do-calculus and standard probability theory Pearl [1995]. These rules define how causal expressions in $\mathcal{L}_{\text{causal}}$ can be transformed while preserving validity under a given causal graph G. Since causal expressions may involve interventions (via do$(\cdot)$), simple syntactic matching is insufficient. Instead, entailment depends on the structure of G and the conditional independencies it encodes.

Do-calculus provides a sound and complete set of transformation rules for this purpose. In our setting, these rules form the basis for reasoning about equivalence between expressions and are central to how we define and implement $\mathcal{F}$.

We now introduce the formal rules that underpin our verification method. These rules form the core component of DoVerifier. Do-calculus consists of three rules that specify when we can remove or add terms to a conditional distribution involving interventions Pearl [1995].

The Rules of do-calculus Let X, Y, Z, and W be arbitrary disjoint sets of nodes in a causal directed acyclic graph (DAG) G [3]. Following the notation of Pearl [2012], we denote:

[3] In do-calculus, X, Y, Z, and W are disjoint sets of variables representing interventions (X), outcomes (Y), observed variables (Z), and other variables (W). These sets can be empty which allows the rules to generalize to many causal inference scenarios.

- $G_{\overline{X}}$ the graph obtained from G by removing all the edges pointing to the nodes in X.
- $G_{\underline{X}}$ the graph obtained by deleting all the edges emerging from the nodes in X.
- $G_{\overline{X}\underline{Z}}$ the graph obtained by deleting edges into X and out of Z.

Each rule applies only if a certain d-seperation condition holds in the modified graph.

Rule 1 (Insertion/deletion of observations):

$$P(y \mid \text{do}(x), z, w) = P(y \mid \text{do}(x), w) \quad \text{if } (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{X}}} \tag{15}$$

This allows us to add or remove observed variables Z from the conditioning set if they are irrelevant to Y once X and W are known (after intervention X).

Rule 2 (Action/observation exchange):

$$P(y \mid \text{do}(x), \text{do}(z), w) = P(y \mid \text{do}(x), z, w) \quad \text{if } (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{X}\underline{Z}}} \tag{16}$$

This allows us to replace an intervention do(Z) with a simple observation, if Z behaves like a non-manipulated variable under this graphical condition.

Rule 3 (Insertion/deletion of actions):

$$P(y \mid \text{do}(x), \text{do}(z), w) = P(y \mid \text{do}(x), w) \quad \text{if } (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{X}\,\overline{Z(W)}}} \tag{17}$$

This allows us to ignore an intervention on Z when it has no causal effect on Y, given the rest of the variables.

Notation: $Z(W)$ is the set of Z-nodes that are *not* ancestors of any W-node in $G_{\overline{X}}$. This restrictions ensures we only remove do-interventions that don't "leak" back into relevant parts of the graph. 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.

$$A \to D, \quad A \to G, \quad B \to F, \quad B \to G,$$
$$C \to E, \quad D \to E, \quad F \to G.$$

C 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. 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

Causal Graph Representation Causal graphs are represented using NetworkX 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

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 $|\pi| < d$ **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

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.

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.

D Proofs

Proposition D.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.

Proof. Let G be a causal DAG with finite node set V. Let $\mathcal{L}_{\text{causal}}$ denote the set of well-formed causal expressions over V, where each expression is of the form $P(Y \mid \mathbf{Z})$ with $Y \subseteq V$ and $\mathbf{Z}$ containing observed or interventional variables (i.e., elements of V or $\mathrm{do}(V)$). Because V is finite, so is the set of possible subsets and intervention/observation combinations, hence $\mathcal{L}_{\text{causal}}$ is countable.

Let $\mathcal{R}$ be the set of valid transformation rules (e.g., the three rules of do-calculus and standard rules of probability). Each rule $r \in \mathcal{R}$ is modeled as a partial function:

$$r : \mathcal{L}_{\text{causal}} \to \mathcal{L}_{\text{causal}}, \tag{18}$$

where $r(E)$ is defined if the syntactic and graphical preconditions (e.g., d-separation in G) for applying r to E are satisfied.

Define the **derivation relation** $\Rightarrow$ on $\mathcal{L}_{\text{causal}}$ by:

$$E \Rightarrow E' \iff \exists r \in \mathcal{R} \text{ such that } r(E) = E'.$$

We now define the derivation graph $S(E_{\text{init}})$ as a directed graph $(\mathcal{V}, \mathcal{E})$, where:

- $\mathcal{V}$ is the set of expressions reachable from E_{init} via a finite sequence of $\Rightarrow$ steps (i.e., derivable expressions);
- $\mathcal{E}$ contains an edge (E, E') if $E \Rightarrow E'$.

To prove the theorem, we must show two things:

(1) Well-definedness. The graph $S(E_{\text{init}})$ is well-defined because:

- Each expression in $\mathcal{L}_{\text{causal}}$ has a canonical syntactic representation.
- Each rule $r \in \mathcal{R}$ is a well-defined partial function whose domain is determined by decidable conditions (syntactic and graphical).
- The derivation relation $\Rightarrow$ is therefore well-defined and finitely composable.

(2) Finite branching. For any node $E \in \mathcal{V}$:

- The number of rule applications is finite, because:
  - The number of rules in $\mathcal{R}$ is finite.
  - Each rule r examines a finite number of subsets of V (e.g., X, Y, Z, W), which are at most $2^{|V|}$ in number.
  - Rules act on bounded-size fragments of expressions and generate outputs in $\mathcal{L}_{\text{causal}}$, which is countable.
- Thus, from any E, only finitely many E' satisfy $E \Rightarrow E'$, i.e., OutDegree(E) is finite.

Hence, $S(E_{\text{init}})$ is a well-defined, finite-branching directed graph. ■

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 D.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`.

First we show that DoVerifier is sound. Suppose we are trying to find a proof sequence starting from E_{init} to E_{target}.

Proof. Assume for contradiction that DoVerifier is not sound. Then there exists some proof path $\pi = \langle E_1, E_2, \ldots, E_k \rangle$ returned by the algorithm such that π is not a valid derivation from E_{init} to E_{target}. This implies that at least one of the following holds:

1. $E_1 \neq E_{\text{init}}$, i.e., the path does not start at the initial expression.
2. $E_k \neq E_{\text{target}}$, i.e., the path does not end at the target expression.
3. There exists some $i \in \{1, \ldots, k-1\}$ such that E_{i+1} is not derivable from E_i via any valid transformation rule admissible under G.

We now show that none of these cases can occur under the design of DoVerifier:

- By construction, the algorithm initializes the search frontier with $\{E_{\text{init}}\}$, so the first element of any returned path is necessarily E_{init}.
- The algorithm terminates only upon finding an expression that is syntactically equal to E_{target}, so $E_k = E_{\text{target}}$.
- The algorithm only expands nodes via valid applications of transformation rules from the set $\mathcal{R}$, which includes do-calculus and standard probability rules. Each edge in the path corresponds to a rule in $\mathcal{R}$, and such rules are only applied if their preconditions (e.g., d-separation) hold in G.

Thus, any returned path must be a valid sequence of derivations from E_{init} to E_{target}, contradicting our assumption. Therefore, DoVerifier is sound. ■
Now we show DoVerifier is complete:

Proof. Suppose $E_{\text{init}} \vdash_G E_{\text{target}}$. Then by definition of $\vdash_G$, there exists a finite sequence of rule applications (i.e., a path in $S(E_{\text{init}})$) from E_{init} to E_{target}. Let the length of this shortest such sequence be d^*. Since $S(E_{\text{init}})$ is a well-defined, finite-branching graph (proposition D.1), BFS explores all nodes reachable from E_{init} up to depth d in increasing order of path length.

Therefore:

- If $d \geq d^*$, then E_{target} will be reached and returned as part of a valid proof sequence.
- If $d < d^*$, then E_{target} is not reachable within the bounded depth, and the algorithm correctly returns `None`.

Thus, the algorithm is complete up to the given depth d. ■
We can further argue that in the case of unbounded depth, if the algorithm terminates upon reaching an expression that is the same as its ancestors, our algorithm is still complete.

Proof. Assume the algorithm is modified so that whenever an expression E is expanded, any successor E' that is already present in the current path π from E_{init} to E is discarded. This prevents cycles. Since $\mathcal{L}_{\text{causal}}$ is **finite** for a fixed set of variables and the rule set $\mathcal{R}$ is finite, the search space $S(E_{\text{init}})$ is also finite. BFS with cycle avoidance will explore all reachable expressions in a finite number of steps. If E_{target} is reachable, BFS will eventually visit it (since BFS is exhaustive in a finite acyclic search space), and return a valid derivation. ■

E Practical Considerations

Fact E.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.

F 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

$$e_1 = P(Y \mid \text{do}(X_1), \ldots, \text{do}(X_k), Z_1, \ldots, Z_m) \tag{19}$$

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:

$$e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} \ldots \xrightarrow{r_n} e_{n+1} \tag{20}$$

where each $r_i \in \{\text{Rule 15, Rule 16, Rule 17}\} \cup \mathcal{P}$. 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 $E_{\text{target}} = e_{n+1}$.

The mean number of edges is 7 (min. 3, max. 10). Rule 15 was used 21172 times, rule 16 was used 29563 times, and rule 17 was used 22508 times.

F.1 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 15, Rule 16, Rule 17) 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.

G 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}
```

H Frequently asked questions

What problem does DoVerifier actually solve? DoVerifier addresses the gap between surface-form evaluation of causal reasoning in LLM outputs (e.g., string match, BLEU, BERTScore) and semantic correctness under causal inference rules. It checks whether a model's predicted causal expression is formally derivable from a given causal graph using do-calculus and probability rules, recovering correct answers that naive metrics miss.

Does DoVerifier require the ground truth answer? For evaluation, yes - the framework needs the correct expression to compare against. However, for feedback and self-correction, it can operate without the ground truth by checking the model's answer against the DAG and suggesting corrections.

Can't we just use the ID algorithm by Shpitser and Pearl [2008] to see if both are identifiable, and then compare? There are cases when the expressions are unidentifiable but can be simplified such as

$$E_1 = P(Y \mid \mathrm{do}(X), \mathrm{do}(W), Z)$$
$$E_2 = P(Y \mid \mathrm{do}(X), Z)$$

under specific DAGs, which can be easily constructed to satisfy do-calculus rule 3.

How is DoVerifier different from Lean or other proof assistants? Lean is a general-purpose formal proof assistant used to verify mathematical theorems in a wide range of domains. It requires users to construct complete proofs in a formal language. DoVerifier is a domain-specific verifier for causal inference. It operates only on causal expressions, uses a fixed set of rules from do-calculus and probability theory, and performs automated proof search to determine equivalence between expressions given a causal graph. Users do not supply the proof steps; the system infers them automatically.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We claim we improve verification on causal statements generated by LLMs we provide a sound and complete algorithm that does so, proves it correct and shows emprical improvements.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [No]

   Justification: Main limitation is due to a lack of dataset, CLadder being the only one that provides a natural question, formal form, and DAG.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: They are proved in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide library used, and details of a pseudocode, the core idea is BFS, which can be easily implemented.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We specify our data generation process.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [NA]

   Justification: No training.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: Not applicable to these type of results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the models used, the computer resources is irrelevant.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: N/A.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: No societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We are transparent on how we prompt the pretrained models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: We do not plan to release code during the review phase.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: All details provided in appendix.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: N/A.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: N/A.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [No]

Justification: We only used it to improve the writing of the paper.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.